НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютеризовані системи управління»
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
на тему: «Модель автоматичної системи керування автомобілем»

Виконав:
студент IV курсу, групи ІА-61
Семченко Ярослав Олександрович _____

Керівник:
старший викладач
Шимкович Володимир Миколайович _____

Рецензент:
кандидат технічних наук, доцент кафедри ОТ
Волокита Артем Миколайович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»
Освітньо-професійна програма «Комп'ютеризовані системи управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проєкт студенту
Семченку Ярославу Олександровичу

1. Тема проєкту «*Модель автоматичної системи керування автомобілем*», керівник проєкту *старший викладач Шимкович Володимир Миколайович*, затверджені наказом по університету від «*07*» *травня* 20*20* р. № *1081-с*.

2. Термін подання студентом проєкту *9 червня 2020 р.*

3. Вихідні дані до проєкту

Швидкість руху до 2 м/с, кут огляду перешкод 30°, поворот коліс за схемою Аккермана, живлення від 4 батарей типу AA (6 В), рух та поворот за допомогою двигунів постійного струму.

4. Зміст пояснювальної записки

Провести аналіз алгоритмів визначення перешкод та руху з урахуванням перешкод; дослідити конструкцію пристрою; описати математичну модель пристрою;

розробити електричну схему пристрою; обрати компоненти пристрою; провести складання пристрою; дослідити алгоритми позиціювання; розробити алгоритм визначення позиції; розробити алгоритм виявлення перешкод; розробити алгоритм руху з урахуванням перешкод; розробити архітектуру програмної системи; розробити програмне забезпечення для роботи з давачами та виконавчими пристроями; імплементувати розроблені алгоритми та перевірити їхню працездатність.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

Схема електрична структурна; схема електрична принципова; діаграма структури програми; блок-схеми алгоритмів; діаграма класів системи.

7. Дата видачі завдання *18 лютого 2020 р.*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Дослідження механічної конструкції	27 березня 2020 р.	
2	Аналіз наявних рішень та вибір алгоритмів	10 квітня 2020 р.	
3	Вибір компонентів, розробка схеми електричної принципової	17 квітня 2020 р.	
4	Збірка пристрою	22 квітня 2020 р.	
5	Розробка програмної системи	8 травня 2020 р.	
6	Імплементація алгоритмів	22 травня 2020 р.	
7	Оформлення текстової документації	1 червня 2020 р.	
8	Розробка графічної конструкторської документації	9 червня 2020 р.	

Студент Ярослав Семченко

Керівник Володимир Шимкович

АНОТАЦІЯ

Семченко Я. О. Модель автоматичної системи керування автомобілем. КПІ ім. Ігоря Сікорського, Київ, 2020.

Проєкт містить 77 сторінок тексту, 46 рисунків, 2 таблиці, посилання на 81 літературних джерела та 21 конструкторський документ.

Ключові слова: безпілотний автомобіль, робот, модель автомобіля, система керування, система, виявлення перешкод, уникнення перешкод, автономний рух.

Об’єктом розробки є система керування моделлю автомобіля.

Мета розробки — створення алгоритмів автоматичного керування моделлю автомобіля для руху з уникненням перешкод.

У дипломному проєкті створено модель автомобіля, розроблено апаратну та програмну частину системи керування нею, а саме: електричну схему та плату, систему керування моделлю, що включає підсистему керування мікроконтролером та давачами, алгоритми обробки даних давачів, визначення позиції, побудови карти перешкод та маршруту руху з уникненням перешкод, підсистему передачі телеметричних даних та програму для їхньої обробки на ПК. Проведено аналіз та вибір оптимального за ціною та простотою методу виявлення перешкод, ґрунтовне дослідження наявних та розробку власних алгоритмів позиціювання, виявлення, визначення перешкод та планування маршруту.

Отримані результати можуть бути використані при розробці мобільних роботів, безпілотних автомобілів, інших рухомих приладів.

SUMMARY

Semchenko Y. O. Car model automatic control system. Igor Sikorsky KPI, Kyiv, 2020.

The project contains 77 pages text, 46 figures, 2 tables, links to 81 literary sources and 21 design documents.

Keywords: self-driving car, mobile robot, car model, control system, obstacle detection, obstacle avoidance, autonomous motion.

Development object is the control system of a car model.

The purpose of the development – creating algorithms for automatic control of a car model for motion with obstacle avoidance.

In the graduation project, a car model was developed, as well as hardware and software parts of its control system, which includes electronic schematic and software system, that consists of: drivers of microcontroller modules and external sensors, algorithms for sensor data processing, positioning, obstacle mapping and obstacle-free path planning, subsystem of navigational data transmission and program for its processing on PC. Different methods of obstacle detection were analyzed, and the one that requires minimum number of low cost components was selected. Existing positioning, obstacle detection and navigational algorithms were researched and improved algorithms were presented.

Obtained results can be applied in development of mobile robots, self-driving cars and similar electronic devices.

№ рядка	Формат	Позначення	Найменування	К-ть арк.	№ екз.	Примітка
			Документація загальна			
			Заново розроблена			
1	А4	ІА61.020БАК.005 ПЗ	Пояснювальна записка	77		
2	А3	ІА61.020БАК.005 Э1	Схема електрична	1		
			структурна			
3	А3	ІА61.020БАК.005 Э3.1	Плата мікроконтролера.	1		
			Схема електрична			
			принципова			
4	А3	ІА61.020БАК.005 Э3.2	Плата двигунів та	1		
			живлення. Схема електрична			
			принципова			
5	А3	ІА61.020БАК.005 Д1	Структура програми	1		
6	А3	ІА61.020БАК.005 Д2	Алгоритм завдання Position	1		
7	А3	ІА61.020БАК.005 Д3	Алгоритм калібрування	1		
			акселерометра			
8	А3	ІА61.020БАК.005 Д4	Алгоритм позиціювання	2		
9	А3	ІА61.020БАК.005 Д5	Алгоритм завдання	1		
			DataProcessor			
10	А3	ІА61.020БАК.005 Д6	Алгоритм оновлення	1		
			карти перешкод			
11	А3	ІА61.020БАК.005 Д7	Алгоритм завдання WiFi	1		
12	А3	ІА61.020БАК.005 Д8	Алгоритм обробки даних	1		
			модуля ESP8266EX			
13	А3	ІА61.020БАК.005 Д9	Алгоритм обробки	1		
			повідомлення AT			

Зм.	Арку	№ докум.	Підпис	Дата	ІА61.020БАК.005 ТП		
Розробив		Семченко Я.О.			Модель автоматичної системи керування автомобілем Відомість	Літера	Аркуш	Аркушів
Перевірив		Шимкович				Т	1	2
Н. Контр.						КПІ ім. І. Сікорського ФІОТ АУТС ІА-61		
Затвердив								

№ рядка	Формат	Позначення	Найменування	К-ть арк.	№ екз.	Примітка
14	А3	ІА61.020БАК.005 Д10	Алгоритм обробки команд	1		
			завдання WiFi			
15	А3	ІА61.020БАК.005 Д11	Алгоритм завдання Log	1		
16	А3	ІА61.020БАК.005 Д12	Алгоритм завдання Driver	1		
17	А3	ІА61.020БАК.005 Д13	Алгоритм завдання	1		
			PathPlanner			
18	А3	ІА61.020БАК.005 Д14	Алгоритм побудови шляху	1		
19	А3	ІА61.020БАК.005 Д15	Діаграма станів драйвера I^2C	1		
20	А3	ІА61.020БАК.005 Д16	Діаграма успадкуванння	1		
			завдань			
21	А3	ІА61.020БАК.005 Д17	Діаграма класів	1		
22	А3	ІА61.020БАК.005 Д18	Програма для ПК. Діаграма	1		
			класів			

№ рядка	Формат	Позначення	Найменування	К-ть арк.	№ екз.	Примітка

Пояснювальна записка

до дипломного проєкту

на тему: «Модель автоматичної системи керування автомобілем»

Київ – 2020 року

ЗМІСТ

Вступ.....5
1 Призначення та галузь застосування 7
2 Огляд наявних рішень.....8
2.1 Огляд методів виявлення перешкод 8
2.1.1 Визначення перешкод за допомогою відеокамери 9
2.1.2 Визначення перешкод за допомогою активних сенсорів 10
2.1.3 Обґрунтування вибору рішення 11
2.2 Алгоритми визначення перешкод.....12
2.3 Огляд алгоритмів планування маршруту.....14
2.3.1 Алгоритми типу Bug 16
2.3.1.1 Перші алгоритми типу Bug.....16
2.3.1.2 Алгоритми типу Bug з давачем відстані 17
2.3.2 Алгоритми потенційного поля 19
2.3.3 Алгоритми на графах 21
2.3.4 Обґрунтування вибору алгоритму.....23
3 Конструкція пристрою.....25
3.1 Загальний вигляд пристрою 25
3.2 Математична модель пристрою.....25
3.3 Фізичне розташування давачів 27
4 Розробка принципової схеми та опис вузлів 28
4.1 Структура пристрою 28
4.2 Мікроконтролер 29
4.3 Давач відстані.....30
4.4 Давач положення.....31
4.5 Модуль Wi-Fi.....31

4.6 Схема під’єднання двигунів 32
5 Розробка алгоритмів 33
5.1 Алгоритми позиціювання 33
5.1.1 Визначення лінійної швидкости руху 33
5.1.1.1 Обробка даних акселерометра 33
5.1.1.2 Обробка даних швидкости 35
5.1.2 Визначення кутової швидкости при повороті 37
5.1.2.1 Визначення кута повороту за допомогою магнітометра 37
5.1.2.2 Визначення кутової швидкости з даних прискорення 39
5.1.3 Опис алгоритму 44
5.2 Алгоритм виявлення перешкод 45
5.2.1 Математична модель давача відстані 45
5.2.2 Опис алгоритму 48
5.3 Алгоритм планування маршруту 49
5.3.1 Розробка алгоритму 49
5.3.2 Опис алгоритму 51
5.4 Алгоритм руху 52
5.4.1 Регулювання швидкости руху 52
5.4.2 Опис алгоритму 53
6 Розробка програмної системи 55
6.1 Система керування автомобілем 55
6.1.1 Структура системи керування 55
6.1.2 Драйвери периферійних пристроїв 58
6.1.2.1 Драйвер ШІМ 58
6.1.2.2 Драйвер двигуна 59
6.1.2.3 Драйвер протоколу I^2C 60

6.1.2.4 Драйвер давача відстані 61
6.1.2.5 Драйвер UART 63
6.1.3 Структура завдань операційної системи 65
6.1.3.1 Драйвер WiFi 65
6.1.3.2 Завдання Log 66
6.1.3.3 Завдання Position 66
6.1.3.4 Завдання DataProcessor 68
6.1.3.5 Завдання Driver 68
6.1.3.6 Завдання PathPlanner 69
6.2 Програма керування для ПК 69
6.2.1 Структура програми 70
6.2.2 Вигляд програми 70
6.3 Протокол обміну даними 72
Висновки 73
Список використаних джерел 75

1 ВСТУП

В кінці XIX – на початку XX століття у світі почали набувати все більшого поширення рухомі пристрої, що не потребують людського управління — роботи, безпілотні автомобілі, літальні апарати. Цілями таких розробок є автоматизація та спрощення людської праці, заміна працівників певних професій автоматичними системами, виконання невиконуваних раніше задач або значне здешевлення задач, що раніше виконувались людьми, що надає можливості масового використання недоступних раніше засобів. Зараз роботи використовуються для вирішення значної кількости прикладних задач від домашнього господарства (наприклад роботи-пилосмоки) до складних промислових підприємств, медицини та безпілотних автомобілів.

Значне поширення та різноманітність задач, де використовуються робототехнічні системи, зумовлює різноманітність їхньої конструкції, можливостей та програмного забезпечення. В галузі систем керування автомобілями об’єкт автоматизації має однакову механіку — чотириколісний автомобіль з поворотом передніх коліс за схемою Аккермана або подібною до неї. На противагу, в галузі малих роботів наявні різні механічні рішення, від вже згаданої поворотної схеми Аккермана до три-, чотири- або шестиколісного робота, де замість поворотних механізмів використовуються окремі двигуни для правих та лівих коліс.

Виходячи з наведеного, для виконання дипломного проєкту було обрано дослідження моделі автомобіля, тобто малого робота з поворотом передніх коліс за схемою Аккермана подібно до автомобілів, та розробку автоматичної системи для керування ним з можливістю виявлення та уникнення перешкод в реальному часі. Для виконання такої задачі необхідно дослідити або розробити ряд алгоритмів — алгоритм визначення поточної позиції відносно початкової, алгоритм виявлення перешкод, алгоритм руху та алгоритм планування маршруту. Окрім того, необхідно забезпечити зв’язок та передачу телеметричних даних на ПК та отримання інформації щодо поставлених користувачем завдань. Було зроблено акцент на створення

алгоритмів, що дозволяють навігацію за допомогою широковживаних та недорогих давачів для наступного використання отриманих результатів в продукції масового вжитку.

Розроблені алгоритми, зокрема алгоритм позиціювання, виявлення перешкод та руху можуть бути застосовані щонайменше у двох сферах — у сфері розробки систем керування безпілотними автомобілями та у сфері розробки малих роботів для виконання прикладних задач. Алгоритми, представлені в даному проєкті, випробовувались та розроблялись для реального пристрою, що означає їхню успішну роботу з практичними, а не теоретичними вхідними даними давачів.

Пояснювальна записка складається з наступних розділів: вступ, призначення та галузь застосування, огляд наявних рішень, конструкція пристрою, розробка принципової схеми та опис вузлів, розробка алгоритмів, розробка програмної системи, висновки, список використаних джерел із 81 найменувань. Обсяг пояснювальної записки 77 сторінок. Графічна частина включає 21 кресленик на 22 аркушах формату А3.

2 ПРИЗНАЧЕННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Метою дипломного проєкту є розробка моделі безпілотного автомобіля та створення автоматичної системи керування ним. Це включає розробку електричної схеми, плати, підбір периферійних пристроїв, розробку драйверів для роботи з ними, а також дослідження, розробку та імплементацію кількох класів алгоритмів:

- алгоритми позиціювання за допомогою різних типів давачів, таких як давач прискорення (акселерометр), гіроскоп, електронний компас (магнітометр);
- алгоритми виявлення перешкод за допомогою різних типів давачів, зокрема камер, ультразвукових, інфрачервоних давачів, лазерів;
- алгоритми побудови моделі навколишнього середовища з використанням наявних даних, отриманих із давачів;
- алгоритми уникнення перешкод та локального прокладання маршруту;
- алгоритми керування двигунами та руху відповідно до прокладеного маршруту.

Розроблена модель автомобіля та система керування ним при додаванні додаткових пристроїв для виконання конкретних завдань може бути застосована при створенні домашніх чи інших прикладних роботів, наприклад роботів-пилосмоків, роботів-офіціантів та інших. Проведені дослідження та розроблені алгоритми можуть бути застосовані також в інших класах рухомих об’єктів автоматизації, таких як безпілотні автомобілі, літальні апарати, рухомі роботи на виробничих підприємствах та навіть роботи-дослідники космічних об’єктів.

3 ОГЛЯД НАЯВНИХ РІШЕНЬ

3.1 Огляд методів виявлення перешкод

Рух з уникненням перешкод є важливою і необхідною задачею в робототехніці, зокрема в розробці систем керування та руху різних типів рухомих автономних систем, від роботів до автомобілів та безпілотних літальних апаратів. Використовуючи різні типи сенсорів та алгоритмів, система керування рухомим пристроєм отримує інформацію про навколишнє середовище та коригує власну поведінку, що дозволяє виконувати рух за заданим маршрутом з урахуванням змін в навколишньому середовищі. Види сенсорів, що найчастіше застосовуються для такого аналізу, та їхній розподіл за категоріями показано на рисунку 2.1. Зокрема, найпоширенішими методами визначення перешкод є активні сенсори — різні типи лазерних, ультразвукових давачів відстані та пасивні сенсори, наприклад відеокамери [79].

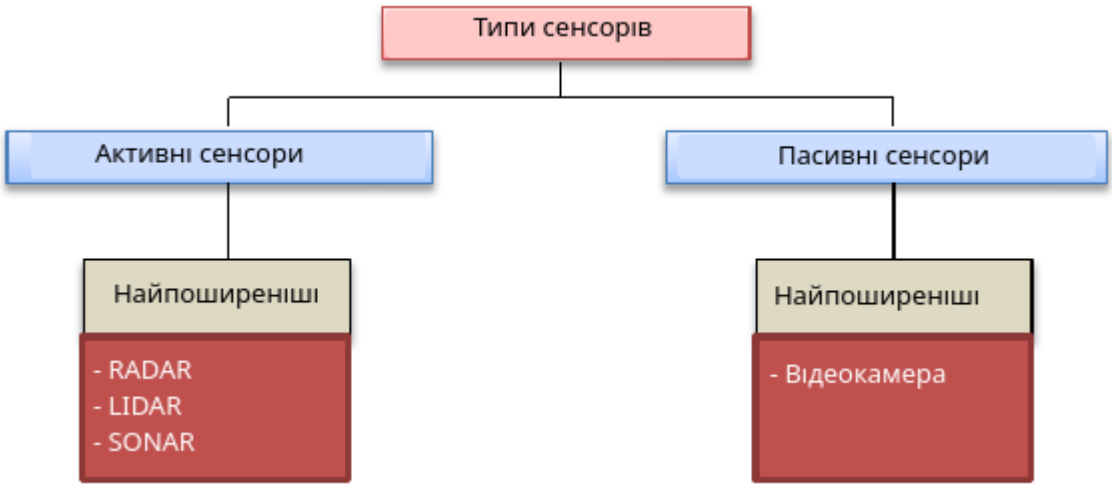


Рисунок 2.1 — Типи сенсорів, що застосовуються для ви*явлення* перешкод

3.1.1 Визначення перешкод за допомогою відеокамери

При визначенні перешкод за допомогою відеокамер зазвичай застосовується одна або дві камери. При використанні однієї камери застосовуються наступні алгоритми:

- алгоритми, засновані на появі відомих об’єктів, базуються на налаштуванні алгоритмів для розпізнавання наперед відомих параметрів об’єктів та оцінки відстані до них на основі розміру даного об’єкту. Такі алгоритми просто реалізуються, ефективно визначають відомі об’єкти, але абсолютно неефективні в умовах невизначеності перешкод;

- алгоритми, засновані на русі, аналізують послідовність зображень та розраховують зміщення кожного пікселя. Маючи інформацію про рух системи, можна розрізнити перешкоди, що виникають на шляху, не маючи попередньої інформації про їхній тип чи форму. Найчастіше для імплементації даного методу використовують алгоритм оптичного потоку.

При використанні стереобачення, тобто поєднанні інформації з двох відеокамер, застосовуються наступні алгоритми:

- метод стереоспівставлення, що базується на знаходженні спільних шаблонів двох зображень, розрахунку відмінностей у формі карти бінокулярних невідповідностей та оцінки відстані до перешкоди на основі горизонтального зміщення;

- метод гомографічної трансформації, що полягає в перетворенні ракурсу зображення, знятого однією камерою, до ракурсу іншої камери, після чого перешкодою вважається кожна значна відмінність між перетвореним зображенням та фактичним зображенням іншої камери.

Зазвичай використовуються комбінації наведених вище методів, що призводить до достатньої якості інформації про навколишній світ та перешкоди. Така інформація може бути використана в складних алгоритмах планування маршруту, зокрема в системах безпілотних автомобілів, що активно розробляються в світі.

До значних недоліків методів визначення перешкод за допомогою відеокамер є їхня висока вартість та велика обчислювальна складність моделей, що вимагає використання нейромереж та високопродуктивних обчислювальних пристроїв, які, у свою чергу, також мають високу вартість та енергозатратність, що особливо важливо при розробці мобільних рухомих пристроїв. [79, 79]

3.1.2 Визначення перешкод за допомогою активних сенсорів

До активних сенсорів відносяться сенсори, які отримують відбитий сигнал після попереднього опромінення навколишнього середовища та обчислюють параметри перешкод на основі прийнятого сигналу. До таких сенсорів відносяться LIDAR (Light Detection and Ranging), RADAR (Radio Detection and Ranging), що використовує радіохвилі, ультразвукові, інфрачервоні сенсори та інші.

- LIDAR використовує підсвічення цілі лазерним випромінюванням для визначення відстані до неї;

- RADAR використовує радіохвилі для визначення кута, типу, відстані та швидкості перешкод;

- ультразвукові сенсори використовують високочастотне звукове випромінення, та аналізують час повернення звуку для визначення відстані до об’єкта.

Перевагами цих методів у порівнянні з відеокамерами є можливість розрізняти об’єкти та перешкоди за різних умов освітлення, дальність, специфічність та точність визначення перешкод. Також обчислювальне навантаження для розрахунку параметрів перешкод є набагато меншим, оскільки усувається необхідність обробляти зображення та видобувати необхідну інформацію з нього. Разом із тим, в кожного з наведених методів є свої переваги та недоліки. Основні переваги та недоліки кожного методу наведені в таблиці 3.1 [79, 79, 79].

Таблиця 2.1 — Порівняння методів визначення перешкод

Можливості	**LIDAR**	**RADAR**	**Ультразвуковий**	**Відеокамера**
Визначення близьких об’єктів	Погане	Гарне	Дуже гарне	Погане
Кут огляду	360°	360°	30°	~90°
Ефективна відстань	~100 м	~0,15 – 250 м	0,03 – 10 м	~250 м
Робота в темряві	+	+	+	-
Визначення швидкости	+	+	-	-
Ціна приладу	$70,000	~$200	$1/шт.	~$100
Ціна обчислювального пристрою	$100+	$100+	$10	$100+

3.1.3 Обґрунтування вибору рішення

З таблиці 3.1 очевидно, що для моделей та малих роботів використання давачів типу LIDAR є невиправданим. Саме тому в більшості промислових прикладних приладів застосовуються RADAR або відеокамери. Однак, ціна таких приладів може бути зависокою для простих застосувань, які вимагають максимального здешевлення компонентів. В таких випадках застосовуються дешевші інфрачервоні та ультразвукові давачі. Порівняння ефективності визначення перешкод за допомогою останніх двох типів давачів досліджено в [79].

В дипломному проєкті вирішено дослідити методи визначення та уникнення перешкод на основі ультразвукового давача відстані через його низьку ціну, простоту роботи та кращу, у порівнянні з інфрачервоним сенсором, якість визначення типових перешкод. Разом із тим, використання такого типу сенсора породжує значні виклики у зв’язку з нетривіальністю алгоритмів обробки даних та керування.

3.2 Алгоритми визначення перешкод

Пересування рухомого робота в змінному навколишньому середовищі можливе лише якщо робот вміє змінювати свою поведінку відповідно до отриманої інформації про зміну навколишнього середовища. В більшости випадків планувальник траєкторії, маючи попередній опис навколишнього середовища, генерує можливі шляхи руху до заданої точки призначення. Використовуючи дані давачів в реальному часі, поведінка робота коригується для уникнення зіткнень за допомогою відповідного перепланування траєкторії руху. Використання ультразвукового давача відстані як основного давача інформації щодо перешкод створює проблеми, оскільки необроблені дані щодо відстані не дають жодної інформації щодо точного місця розташування та типу перешкоди. Приклад такої ситуації показано на рисунку 2.2 [79].

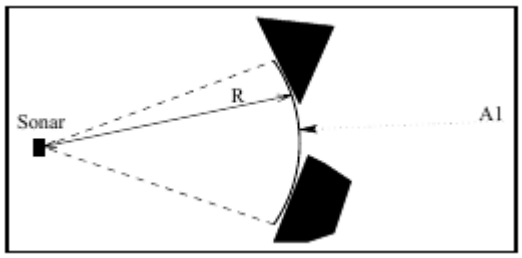


Рисунок 2.2 – Дві перешкоди, що дають однакові *виміри*

Таким чином, повна інформація щодо перешкод може бути отримана лише в разі огляду перешкоди з різних точок. Оскільки робот виконує визначену задачу, повний об'їзд перешкоди з усіх боків є неможливим. Система повинна отримати максимально можливу інформацію з отриманих під час руху даних. Оскільки при русі автомобіля з кожним новим виміром давача відстані з'являється інформація щодо інших точок навколишнього середовища, необхідно зберігати та поєднувати інформацію з послідовних вимірів в неперервну мапу навколишнього середовища.

Використання моделювання навколишнього середовища за допомогою накладання графічних примітивів (ліній, багатокутників, кіл та ін.) вимагає наявність інформації про навколишнє середовище з високою роздільною здатністю, а також великий обсяг попередньої обробки даних. Навіть у такому випадку такий підхід

може давати помилкові результати через похибки, помилкові виміри давачів та інші неточності.

Широко використовуваною альтернативою є зображення мапи навколишнього світу за допомогою різних типів матриці заповненості. Розрізняють два види такої матриці:

- растрова матриця, або матриця з фіксованим розміром комірки;
- матриця з адаптивним розміром комірки (наприклад, з використанням дерева квадрантів — приклад наведено на рис. 2.3)

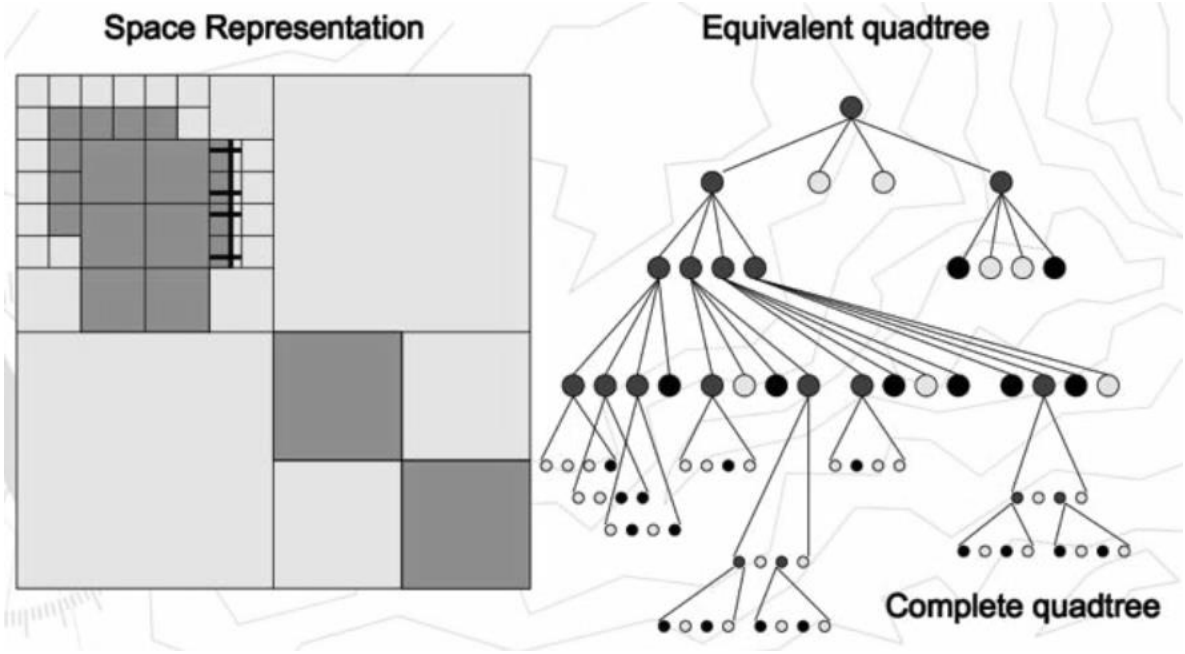


Рисунок 2.3 — Приклад представлення простору за допомогою дерева квадрантів

У кожного з цих видів є свої переваги та недоліки. Використання растрової матриці вимагає використання великого, але фіксованого обсягу пам’яті, тоді як використання адаптивного розміру комірки дозволяє зменшити використання пам’яті в разі, коли перешкоди великі за розміром і займають кілька комірок, або, навпаки, більшість простору незаповнена. Однак, таке розбиття вимагає значного збільшення обчислювального часу при частому оновленні даних щодо навколишніх перешкод, а також ускладнює застосування імовірнісних алгоритмів, в разі застосування яких в комірках зберігається імовірність знаходження перешкоди в даній точці, а не дискретне значення зайнятості комірки.

В даному проєкті навколишній простір представлений за допомогою растрової матриці заповненості. Така матриця є двовимірним масивом, кожна комірка якого відповідає певному проміжку координат навколишнього світу. В комірці зберігається оцінка ймовірности того, що за даними координатами в просторі знаходиться перешкода. Після кожного сканування оновлюються ті комірки, інформація щодо яких з'явилась з останнього виміру [79, 79]. Візуалізація матриці заповненості при проїзді автомобіля між двома перешкодами в кімнаті наведено на рисунку 2.4. Опис алгоритму наведено в розділі 48.

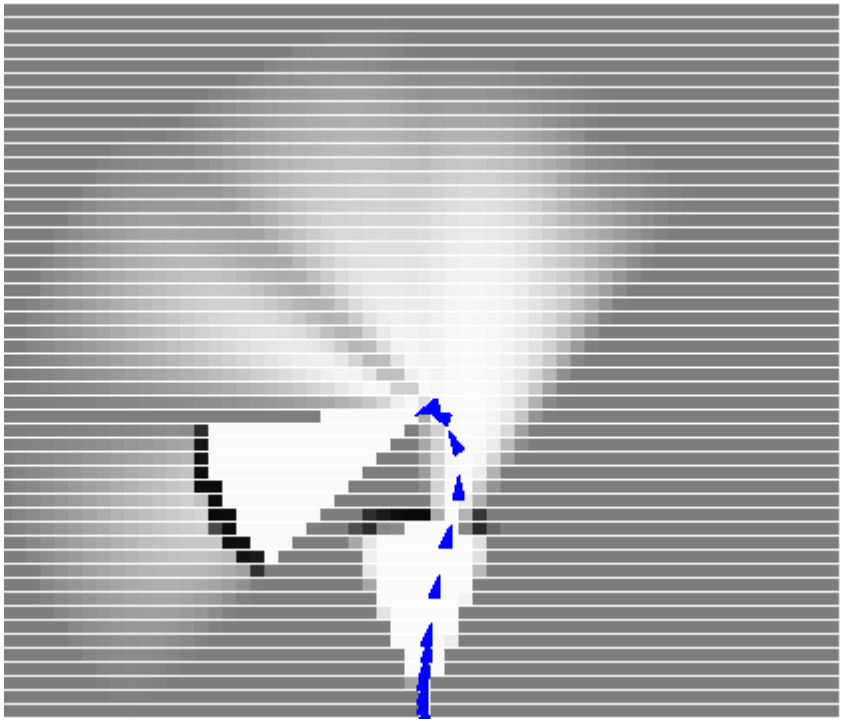


Рисунок 2.4 — Візуалізація матриці заповненості та маршруту руху

3.3 Огляд алгоритмів планування маршруту

Планування маршруту руху робота означає процес пошуку, при наявній (часто неповній) карті перешкод та знаючи позицію цільової точки, геометричного представлення шляху робота від поточної позиції до цілі, рухаючись за яким, він дістанеться до точки призначення без зіткнень[79]. Зазвичай, інформація робота про навколишнє середовище значно обмежена, а іноді відсутня, окрім того, вона має властивість швидко змінюватися (наприклад, при появі рухомих об'єктів бли-

зько до робота). Таким чином, алгоритми планування маршруту для кожного робота мають враховувати цю неточність та неповноту інформації. Залежно від типу робота, середовища та методів руху, для ефективності пошуку маршруту використовуються певні припущення, за якими виконуються спрощення моделі світу.

Наразі розроблено значну кількість алгоритмів уникнення перешкод та планування маршруту, починаючи від простої зупинки пристрою при виявленні перешкоди й закінчуючи адаптивними змінами до поведінки робота залежно від виявлених типів, розмірів та інших характеристик перешкод. Ці алгоритми різняться залежно від кількости даних чи типів сенсорів, необхідних для їхньої роботи, швидкодії, просторової складності, ефективності та стратегії керування [79]. Найпоширеніші алгоритми, що використовуються в сучасній робототехніці, поділяються на такі категорії:

- алгоритми типу Bug:
  - наївні, або найпростіші, алгоритми;
  - алгоритми з використанням давача відстані;
- алгоритми, базовані на потенційному полі;
- алгоритми на графах:
  - формальні алгоритми (алгоритм Дейкстри, Флойда-Воршела);
  - евристичні алгоритми (наприклад пошук в ширину);
  - алгоритми гібридного пошуку (наприклад, A*, D*).

3.3.1 Алгоритми типу Bug

3.3.1.1 Перші алгоритми типу Bug

Алгоритми типу Bug використовують найпростіший, наївний підхід, який полягає в русі прямо до цілі аж до моменту зустрічі з перешкодою. У випадку зустрічі з перешкодою визначається її контур, після чого розраховується подальший

шлях до цілі. Ці алгоритми не використовують збереження минулих даних чи попередньо обчислені карти, використовуються лише безпосередні значення давачів[79]. Перші алгоритми цього класу, Bug-1 та Bug-2, запропоновані в 1986 році В. Люмельським та О. Степановим, відрізняються умовою припинення руху по контуру перешкоди та відновлення руху до цілі.

В алгоритмі Bug-1, щойно виявлено перешкоду, робот виконує повний об'їзд, починаючи з точки H. Повністю визначивши контур перешкоди, робот розраховує точку з найменшою відстанню до цілі, яка називається "точкою відправлення". Після цього він знову рухається по контуру до точки відправлення, при досягненні якої повертає в напрямку цілі. Такий алгоритм є дуже неефективним, проте гарантує досягнення будь-якої цілі, яка може бути досягнена[79, 79]. Приклад руху за алгоритмом наведено на рисунку 2.5[79].

Для збільшення ефективності алгоритму було запропоновано алгоритм Bug-2, в якому робот припиняє рух по контуру, щойно він перетнув "M-лінію", або лінію курсу — умовну лінію між початковим положенням робота та ціллю. Порівняння алгоритмів Bug-1 та Bug-2 наведено на рисунку 2.5[79].

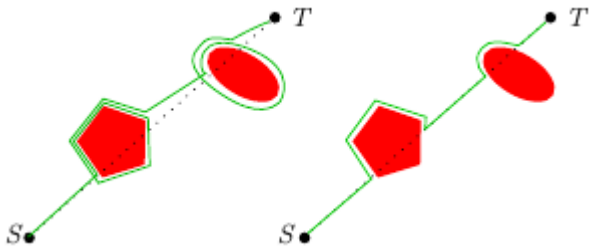
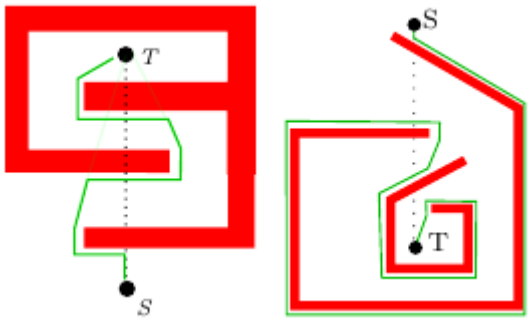


Рисунок 2.5 – Порівняння поведінки алгоритмів Bug-1 (ліворуч) та Bug-2

Простота цих алгоритмів спричиняє кілька недоліків. Перш за все, шлях, пройдений пристроєм не буде оптимальним. По-друге, ці алгоритми не беруть до уваги механічну конфігурацію робота [79], зокрема, виконання такого типу руху є неможливим для автомобіля.

Для усунення першої з цих проблем в 1997 році І. Камон та Е. Рівлін запропонували алгоритм DistBug. Він має дві важливі відмінності від алгоритму Bug-2. По-перше, точкою відправлення вважається точка, відстань від якої до цілі менша, ніж від наступної точки перешкоди. По-друге, на відміну від алгоритмів Bug-1 та Bug-2, де напрямок руху вздовж перешкоди задається наперед, тут напрямок обирається з кута, під яким робот підходить до перешкоди. Хоча в більшости випадків це призводить до значного зменшення відстані, яку необхідно пройти, в деяких випадках цей алгоритм призводить і до її збільшення. Приклад обох варіантів наведено на рисунку 2.6[79].

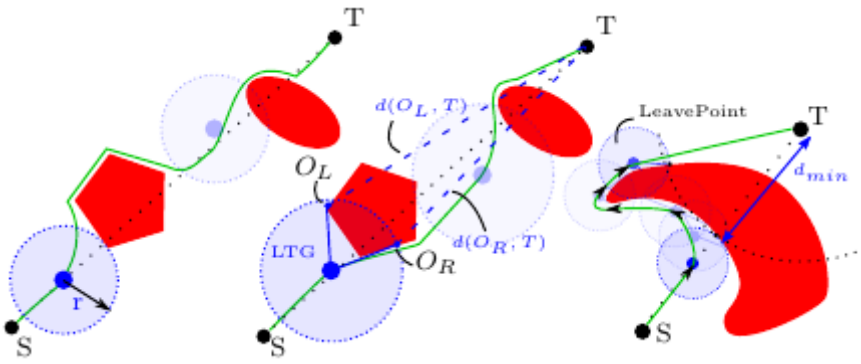


(а) (б)

Рисунок 2.6 – *Покращення (а) та погіршення (б) руху робота за алгоритмом DistBug*

3.3.1.2 Алгоритми типу Bug з давачем відстані

Всі алгоритми, перераховані вище, виходять з припущення, що робот може визначити наявність перешкоди лише в безпосередній близькості до неї. Проте більшість сучасних давачів, що використовуються в роботах та автомобілях, здатні виявляти перешкоди на відстані до них. Виходячи з цих можливостей, Люмельський та Ск'юіс в 1988 та 1990 роках запропонували алгоритми VisBug-21 та VisBug-22. Вони обидва базуються на алгоритмі Bug-2, тобто при русі робот дотримується "М-лінії", проте значно скорочує траєкторію. Приклад роботи алгоритму показано на рисунку 2.7 (а)[79].



(а) (б) (в)

Рисунок 2.7 – *Алгоритми VisBug (а) та TangentBug (б, в)*

Іншим і більш успішним підходом став алгоритм TangentBug, розроблений І. Камоном, Е. Рівліним та Е. Рімоном в 1997 році на основі алгоритму DistBug. Приклад роботи цього алгоритму наведено на рисунку 2.7 (б, в). LTG означає локальний тангенційний граф, який будується в межах досяжності давача відстані, після чого алгоритм шукає точку O_i оминання перешкоди, яка мінімізує очікувану суму відстаней $d(x, O_i) + d(O_i, T)$, де x – поточна позиція робота, а T – цільова точка. На рисунку 2.7 (в) показано випадок, коли початково оптимальний шлях, з отриманням нових даних, виявився значно довшим. В такому випадку алгоритм зберігає оцінку значення $d(O_i, T)$, після чого продовжує рух вздовж перешкоди до моменту, коли відстань до цілі буде меншою за цю оцінку. Після цього робот покидає перешкоду та продовжує рух до цілі.

Цей алгоритм виявився найуспішнішим з сімейства Bug, і багато модифікацій було створено на його основі. Зокрема, алгоритм WedgeBug враховує обмежений кут огляду давача відстані, а алгоритм InsertBug – радіус безпеки робота навколо перешкоди. Загальна класифікація алгоритмів типу Bug наведена на рисунку 2.8[79].

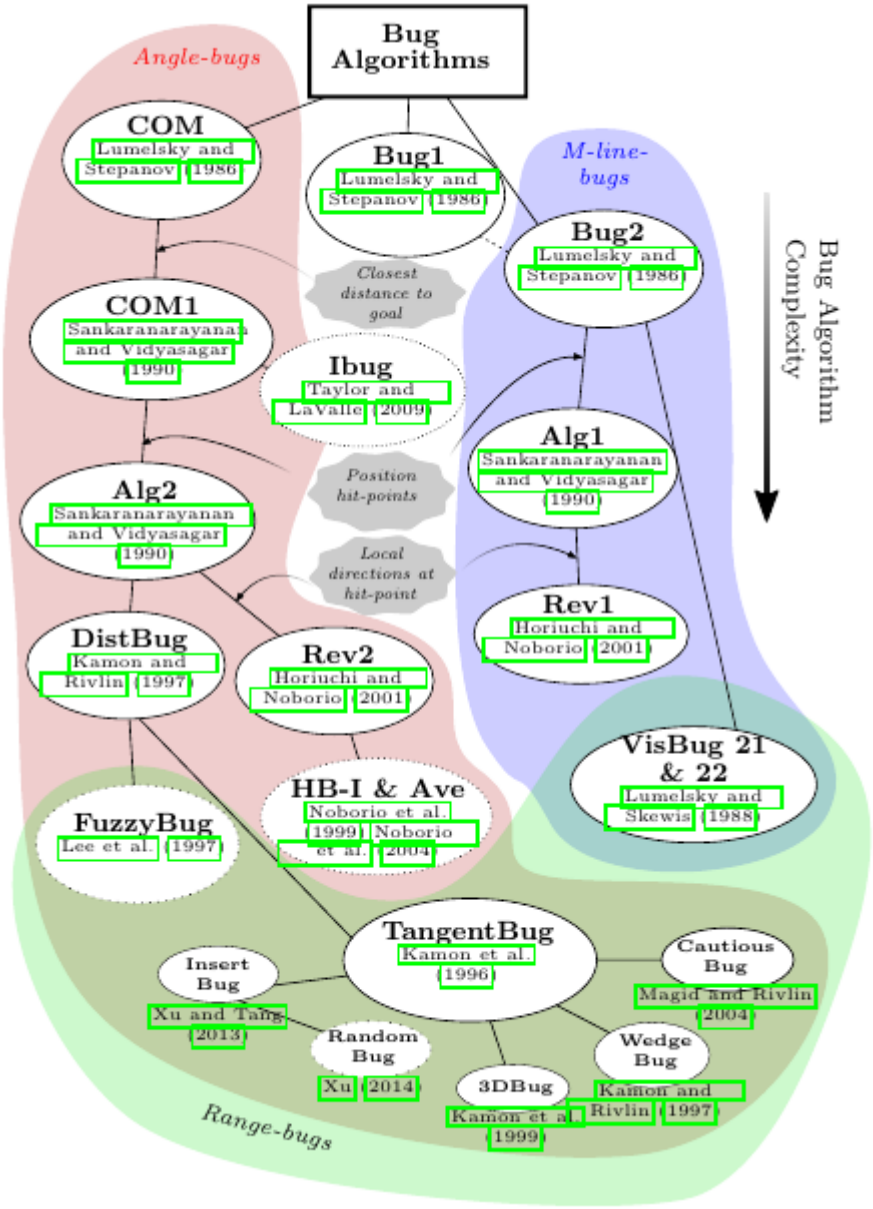


Рисунок 2.8 — Класифікація алгоритмів типу Bug

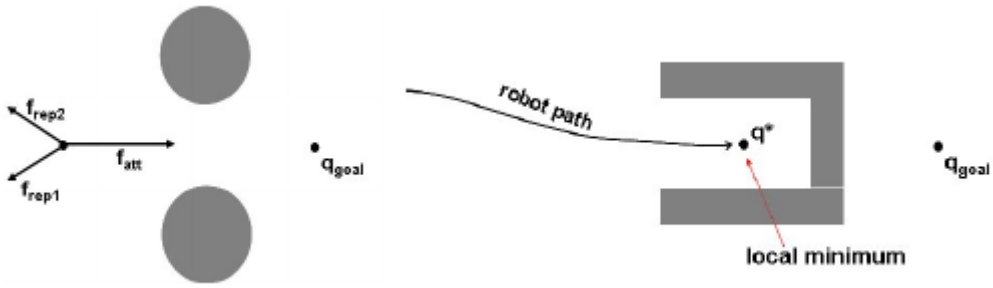
3.3.2 Алгоритми потенційного поля

Вперше алгоритм навігації за допомогою штучного потенційного поля був запропонований у статті О. Хатіба в 1995 році. Згідно з цією концепцією, робот вважається частинкою, яка рухається в потенційному полі, яке генерується точкою призначення та перешкодами. Точка призначення створює потенціал притягування, тоді як перешкоди створюють потенціал відштовхування. Робот рухається в цьому

полі під дією сили, яка притягує його до точки призначення, водночас відштовхуючи від перешкод на шляху [79]. Таке потенційне поле називають полем векторних сил (Vector force field, VFF).

Побудова потенційного поля може виконуватись як із попередньо відомої карти перешкод, так і з отриманих під час руху даних давачів. В другому випадку зміна інформації про поле вимагає перерахунку сил, що діють у всьому полі.

Використання такого підходу має деякі обмеження, які роблять його непридатним до використання без додаткових модифікацій. Наприклад, на рисунку 2.9 показано конфігурації перешкод, за яких робот не досягає точки призначення.



(а) (б)

Рисунок 2.9 – *Випадки перешкод, в яких рух потенційним полем припиняється раніше, ніж ціль досягнуто*

Для уникнення ситуацій, зображених на рисунку 2.9 (б), була запропонована побудова потенційного поля як сума двох компонент, однією з яких є цільове поле, тобто напрямок руху до цільової точки, а другою — загороджувальне поле, яке складається з сил відштовхування від перешкод. Сумою цих компонент є навігаційне поле. Таке поле може бути зображене як схил з горами, де градієнт потенціалу дорівнює нахилу поля. Приклад такого представлення показано на рисунку 2.10[79].

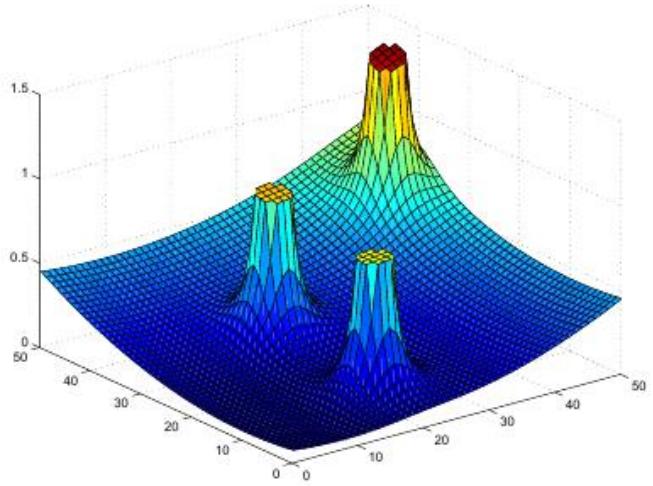


Рисунок 2.10 — Графічне представлення потенційного поля

З усім тим, наведений алгоритм не може уникнути випадків, таких як на рисунку 2.9 (а), тому є непрактичним для руху в середовищах, де виникають замкнені коридори. Для виправлення ситуації застосовуються різні підходи, зокрема нейронні мережі, алгоритми керування вищого рівня, а також вдосконалені алгоритми, потенціальне поле в яких розраховується враховуючи не лише положення робота й перешкод, але й положення робота відносно перешкоди, а також ігноруючи перешкоди, що знаходяться позаду робота.

Значним недоліком також є те, що розрахунок потенційного поля передбачає попереднє знання, нехай і неточне, більшости перешкод, що знаходяться в межах поля. У випадку значної неповноти інформації розраховане потенційне поле буде хибним і, в деяких випадках, може завадити роботу досягнути точки призначення. Іншим недоліком є представлення робота як частинки, що не враховує його кінематичні характеристики (наприклад, автомобіль не може розвернутися на значний кут без проїзду певної відстані.

3.3.3 Алгоритми на графах

Алгоритми, засновані на графах, передбачають розбиття простору на комірки, які вважаються вершинами, та створенні графа шляхом з’єднання суміжних

комірок за допомогою ребер різної ваги. Ребра до вершин, в яких знаходяться перешкоди, зазвичай позначають нескінченно великою вагою, що дозволяє використовувати будь-який алгоритм пошуку в графі.

Використання формальних алгоритмів на кшталт алгоритму Дійкстри чи Флойда-Воршелла не враховує бажаний напрямок руху чи інші наперед відомі параметри середовища, що призводить до великої кількости вершин, які необхідно обробити. Наприклад, алгоритм Дійкстри шукає точку призначення рівномірно в усіх напрямках, а алгоритм Флойда-Воршелла обробляє всі наявні ребра, що призводить до квадратичної складности алгоритму.

У випадку наявности попередньо визначеної евристики (найпростіша — відстань до цілі) може бути застосований евристичний алгоритм пошуку в ширину. Фактично, результатом роботи такого алгоритму буде шлях робота, подібний до алгоритмів типу Bug.

Ідея алгоритму гібридного пошуку A* полягає в застосуванні наявної евристики до алгоритму Дійкстри. Алгоритм A* використовує евристику для визначення вершин, які розглядаються в першу чергу, таким чином скорочуючи час роботи алгоритму Дійкстри шляхом зменшення кількости вершин, які мають бути розглянуті до знаходження шляху. При цьому зберігається властивість алгоритму Дійкстри щодо гарантованого знаходження шляху, а якщо евристика є прийнятною (admissible) — також гарантується знаходження найкоротшого шляху.

Порівняння роботи наведених алгоритмів показано на рисунку 2.11[79]:

1) алгоритм Дійкстри;
2) алгоритм пошуку в ширину;
3) алгоритм A*.

На рисунку зображено вершини, що розглянуті алгоритмами до знаходження повного шляху, та побудований шлях.

Усі алгоритми, перераховані вище, працюють за умови наявності апріорної (попередньої) карти середовища і мають бути запущені з нуля щоразу, коли карта

змінюється. Для покращення ефективності роботи алгоритмів в змінному середовищі було розроблено наступні два сімейства вдосконалень A*, які дозволяють оновлювати розрахований шлях за наявності нової інформації:

- алгоритми D*: D* та Focused D*;
- алгоритми LPA*: LPA* та D* Lite.

Обидва варіанти використовують алгоритм A*, зберігаючи всі його параметри. У випадку, коли наявна інформація про відому частину середовища може застарівати та не оновлюватись, таке збереження передбачає додаткові виклики. Значним недоліком наведених алгоритмів є їхня просторова складність, яка складає b^2, де b – кількість вершин, або комірок матриці перешкод, а також обробка значної кількости комірок при кожному оновленні даних, можливо, по декілька разів.

3.3.4 Обґрунтування вибору алгоритму

Перш за все, з наведеної інформації можна зробити висновок, що алгоритми, базовані на потенційному полі, мають низку недоліків:

- вимагають інформацію про існування всіх перешкод, навіть тих, які знаходяться поза межами дії давачів;
- мають велику обчислювальну складність, оскільки передбачають повний перерахунок потенційного поля з кожним новим виміром;
- не гарантують досягнення цілі при певних конфігураціях перешкод або неповних початкових даних;
- не враховують кінематику робота.

Таким чином, алгоритмами, які можуть бути застосовані в даному проєкті, є одне із вдосконалень A* або TangentBug-базований алгоритм. Також алгоритм, застосований в проєкті, має враховувати кінематичну модель автомобіля.

Оскільки базою для даного проєкту є малопотужний обчислювальний пристрій, а давач відстані дає можливість дізнатися про наявність лише найближчої перешкоди в єдиному напрямку, застосування алгоритму типу A* є неспівмірним з

задачею, яку необхідно виконати. Для досягнення очікуваного результату достатньо алгоритму, який призначений для обробки такого обсягу даних, який наявний виходячи з використаних давачів.

Саме тому для розв'язання задачі планування шляху в даному проєкті за основу був взятий алгоритм TangentBug[80] із частиною вдосконалень, запропонованих для врахування обмеженості кута огляду давача відстані в алгоритмі WedgeBug[80], а також конкретних методів планування маршруту за допомогою векторів з врахуванням "радіусу безпеки" навколо перешкод, запропонованих в алгоритмі InsertBug[80]. Модифікація зазначених алгоритмів враховує кінематику автомобіля та особливості давачів. Алгоритм описаний в розділі 52.

4 КОНСТРУКЦІЯ ПРИСТРОЮ

4.1 Загальний вигляд пристрою

Для механічної частини моделі було використано машинку з радіокеруванням (рис. 3.1). Використано наявні двигуни постійного струму для приведення до руху та повороту, а також батарейний відсік. Для взаємодії з механічною частиною розроблено плату контролю живлення та керування двигунами на біполярних транзисторах за схемою H-Bridge [79]. Схему електричну принципову цієї плати наведено в конструкторській документації на кресленику Э3.2.

Рисунок 3.1 — Вигляд моделі безпілотного автомобіля

4.2 Математична модель пристрою

Як основа математичної моделі автомобіля використано “двоколісну” (bicycle) кінематичну модель, або модель автомобіля з двома ступенями свободи (рис. 3.2). В ній два передніх та два задніх колеса змодельовані одним усередненим

колесом, розташованим на повздовжній осі автомобіля. Таким чином, кути повороту передніх коліс, які відрізняються в разі використання Акерманівського поворотного механізму [79], для цілей визначення положення та напрямку руху усереднені кутом повороту одного “середнього” колеса, зображеного на рисунку як δ.

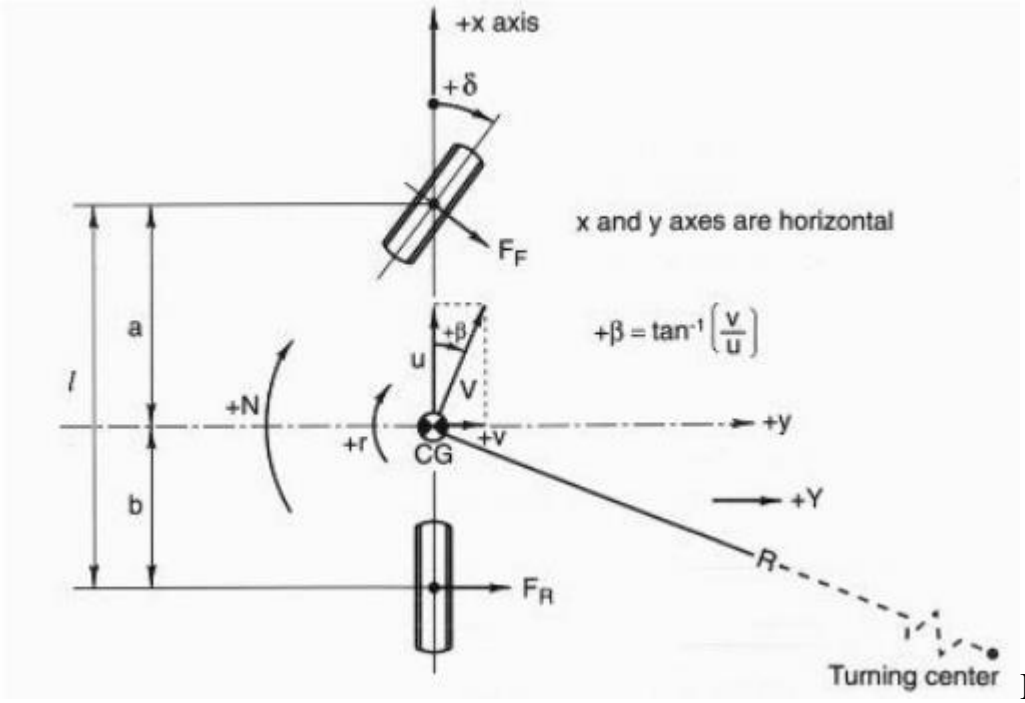


Рисунок 3.2 — Математична модель автомобіля

Умовні позначення на рисунку 3.2:

CG — центр мас автомобіля. За результатами дослідження використаної моделі, її центр мас збігається з серединою міжосьової відстані, тобто *a=b=l/2=70 мм*;

δ — кут повороту передніх коліс;

β — кут повороту корпусу автомобіля;

R — радіус повороту автомобіля.

Тоді, мають місце наступні математичні тотожності [80]:

$$\omega = \frac{V\cos(\beta)\tan(\delta)}{a+b} = \frac{V\cos(\beta)\tan(\delta)}{l}; \quad (3.1)$$

$$\begin{matrix} \dot{X} = V\cos(\alpha+\beta) \\ \dot{Y} = V\sin(\alpha+\beta) \end{matrix}; \quad (3.2)$$

$$\beta = \arctan(\frac{a\tan(\delta)}{a+b}) = \arctan(\frac{\tan(\delta)}{2}), \quad (3.3)$$

де:

ω — кутова швидкість повороту автомобіля;

V — лінійна швидкість руху автомобіля;

Ẋ, Ẏ — зміна положення по осях X та Y за квант часу;

решта позначень відповідають позначенням на рисунку 3.2.

4.3 Фізичне розташування давачів

Модуль акселерометра та магнітометра LSM303C (інерційний модуль) розташовано на осі передніх коліс та на повздовжній осі. Вісь X вирівняна за повздовжньою віссю автомобіля, вісь Y збігається з віссю передніх коліс. Таким чином, можна вважати, що інерційний модуль розташовано на “середньому” передньому колесі моделі.

Ультразвуковий давач відстані розташований на передній частині автомобіля на повздовжній осі. Відстань від осі передніх коліс (та інерційного модуля) до давача відстані становить 20 міліметрів. Давач розташований на висоті, необхідній для уникнення відбиття звукових хвиль від поверхні землі.

5 РОЗРОБКА ПРИНЦИПОВОЇ СХЕМИ ТА ОПИС ВУЗЛІВ

5.1 Структура пристрою

Схему електричну принципову плати мікроконтролера та підключення решти компонентів системи наведено в конструкторській документації на кресленику Э3.1, плати двигунів на кресленику Э3.2. Структурну схему наведено на рисунку 4.1, а також в конструкторській документації на кресленику Э1.

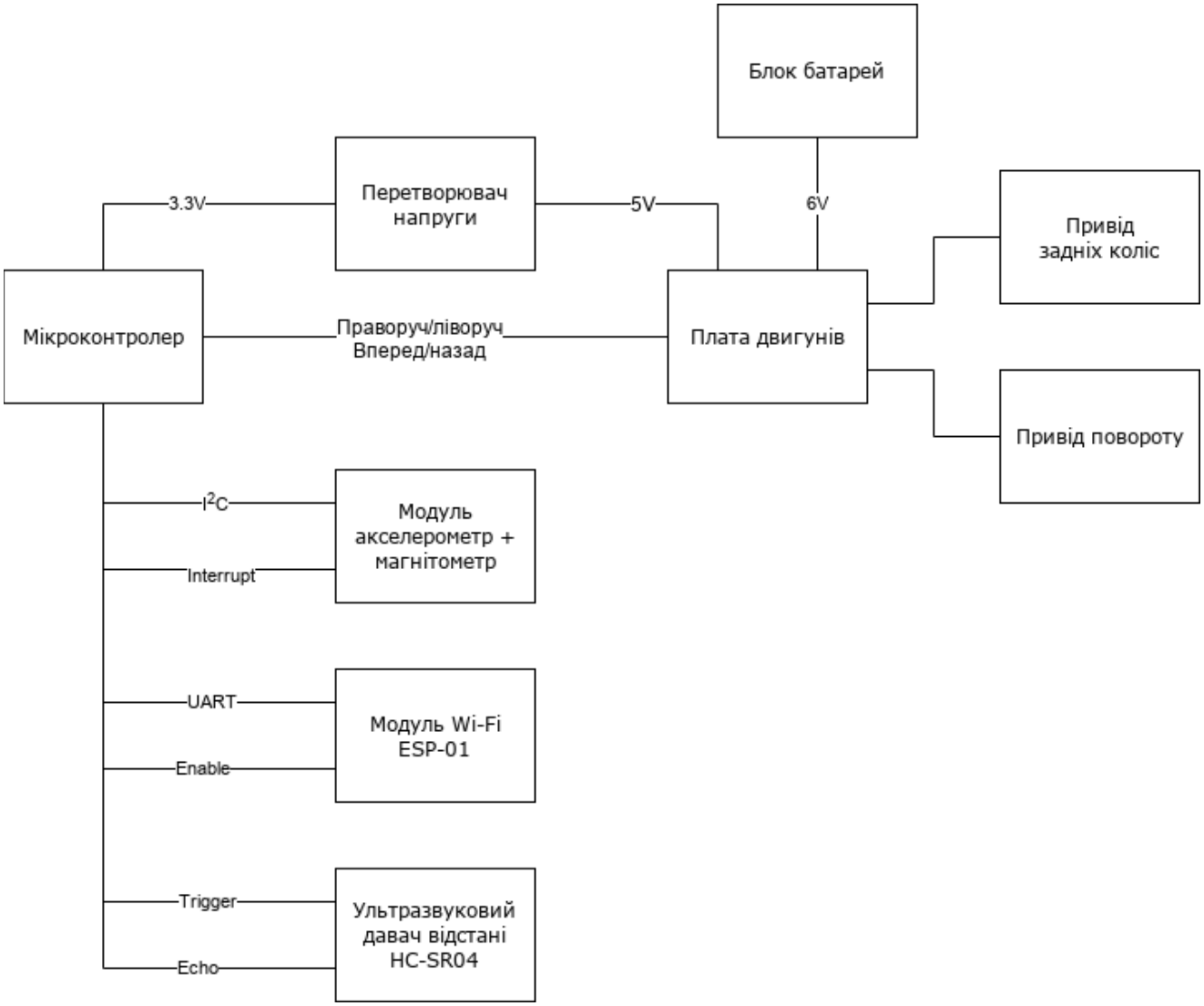


Рисунок 4.1 – Структурна схема пристрою

Живлення відбувається від чотирьох батарей типу LR6 розміру AA, таким чином напруга живлення системи близько 6 В, які подаються на плату двигунів. Вона складається з двох транзисторних схем типу H-Bridge[79], до кожної з яких

під’єднано двигун постійного струму, один для руху по прямій лінії, та інший для повороту передніх коліс. Для приводу задніх коліс до схеми додано “flyback” діоди Шотткі, необхідні для безпечного комутування при використанні ШІМ для керування потужністю двигуна.

Після стабілізації за допомогою конденсаторів, напруга подається на лінійний регулятор з виходом 5 В, звідки подається на плату мікроконтролера. За допомогою регулятора напруги RT9013 отримується живлення 3.3 В, необхідне для мікроконтролера, акселерометра та бездротового модуля. Живлення ультразвукового давача відстані відбувається від напруги 5 вольтів.

5.2 Мікроконтролер

Як обчислювальний пристрій обрано мікроконтролер STM32F401. До нього під'єднано необхідні для роботи конденсатори та індуктивності. Кварцеві резонатори Y1 (25 МГц) та Y2 (32.768 КГц), які під’єднані за типовою схемою з використанням двох конденсаторів, використовуються для тактування мікроконтролера (за допомогою внутрішніх схем тактування та помноження частоти) та блоку годинника реального часу. Також під’єднано кнопку RESET та світлодіод. Додатково під’єднано подільник напруги на резисторах R16, R17, вихід якого з’єднано з АЦП мікроконтролера для контролю рівня напруги на лінії 5 В. Пряме підключення лінії 5 В до мікроконтролера неможливе, оскільки напруга живлення мікроконтролера становить 3.3 Вольти.

Основні характеристики мікроконтролера STM32F401CC[80]:

- 32-бітовий мікропроцесор ARM Cortex-M4 з математичним співпроцесором (FPU);
- 256KB програмованої постійної пам’яті (flash);
- 64KB оперативної пам’яті (RAM);
- робота на частоті до 84 МГц (в даному проєкті використано частоту 80 МГц);

- 11 таймерів з різними типами входів та виходів, підтримкою ШІМ та вхідного захоплення сигналів, а також таймер системного переривання;

- інтерфейси I^2C, USART (універсальний синхронний та асинхронний прийомач-передавач), SPI, SDIO, USB 2.0;

- DMA (контролер прямого доступу до пам’яті) для обміну даними між пам’яттю та периферійними модулями без використання процесорного часу;

- налагоджувальні інтерфейси JTAG та SWD.

Даний мікроконтролер було обрано за сукупністю наступних параметрів:

- наявність модуля обробки чисел з рухомою крапкою;
- достатньо висока продуктивність для обробки значних масивів даних;
- обсяг оперативної пам’яті достатній для зберігання масивів даних ;
- наявність достатньої кількости таймерів, інтерфейсів I^2C та UART.

5.3 Давач відстані

Як ультразвуковий давач відстані обрано HC-SR04, оскільки він найбільш поширений, дешевий та має гарні характеристики [79]. Він під’єднується до мікроконтролера за допомогою двох ліній, позначених Trig та Echo. Лінія Trig використовується для подання на давач сигналу довжиною 10 мікросекунд, після отримання якого він розпочинає випромінення восьми ультразвукових пакетів частотою 40 КГц. Після надсилання сигналу пристрій встановлює лінію Echo в логічну “1”,

яку скидає при отриманні відбиття сигналу або при завершенні часу очікування. Приклад обміну даними показано на рисунку 4.2[80].

Основні характеристики модуля HC-SR04:

- кут ультразвукового променя 30°, тобто розсіювання ±15°;
- точність: 3 мм;
- живлення від напруги 5 В;
- ефективна відстань 20 — 5000 мм;

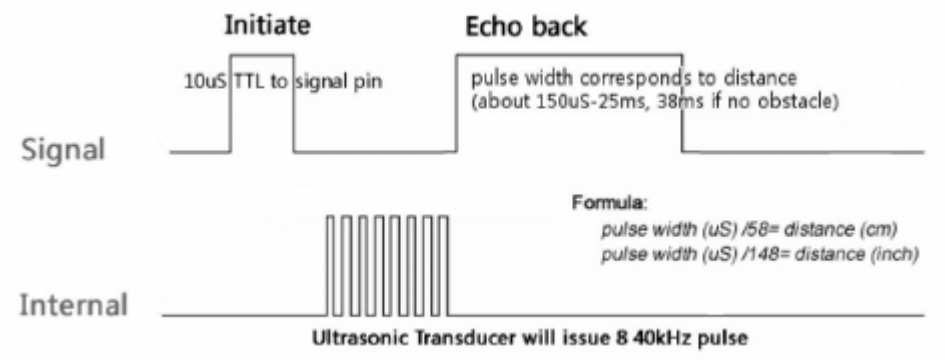


Рисунок 4.2 – Обмін даними між мікроконтролером та ультразвуковим давачем

- споживання струму 10-20 мА.

5.4 Давач положення

Як інерційний давач положення було обрано модуль акселерометра та магнітометра ST LSM303C, який застосовується для визначення швидкости, положення, кута повороту та напрямку руху. Він з'єднується з мікроконтролером за допомогою протоколу передавання даних I^2C (Inter-Integrated Circuit). Інтерфейс описано в [80]. Також виділено дві лінії переривань для сигналізування про наявність нових даних. Принцип взаємодії за протоколом I^2C описано в розділі 63, а протокол взаємодії з пристроєм — в розділі 70.

Основні характеристики модуля LSM303C[80]:

- тривісне вимірювання прискорення та магнітного поля;
- вимірюване магнітне поле до 16 гаусів, прискорення до 8 гравітаційних;
- вбудоване 16-бітове цифрування даних;

- взаємодія за протоколом I^2C або SPI;
- напруга живлення 2-3.6 В;
- вбудований давач температури;
- можливість зберігання до 32 вимірів акселерометра в буфері;
- частота вимірів акселерометра 800 Гц, магнітометра — 80 Гц.

5.5 Модуль Wi-Fi

Для передавання даних за допомогою Wi-Fi обрано модуль ESP-01 з мікроконтролером ESP8266EX. Взаємодія з модулем відбувається за протоколом UART з використанням AT-команд. Увімкнення модуля відбувається за допомогою лінії CH_PD. Опис та алгоритми процесу взаємодії з модулем описані в розділі 68.

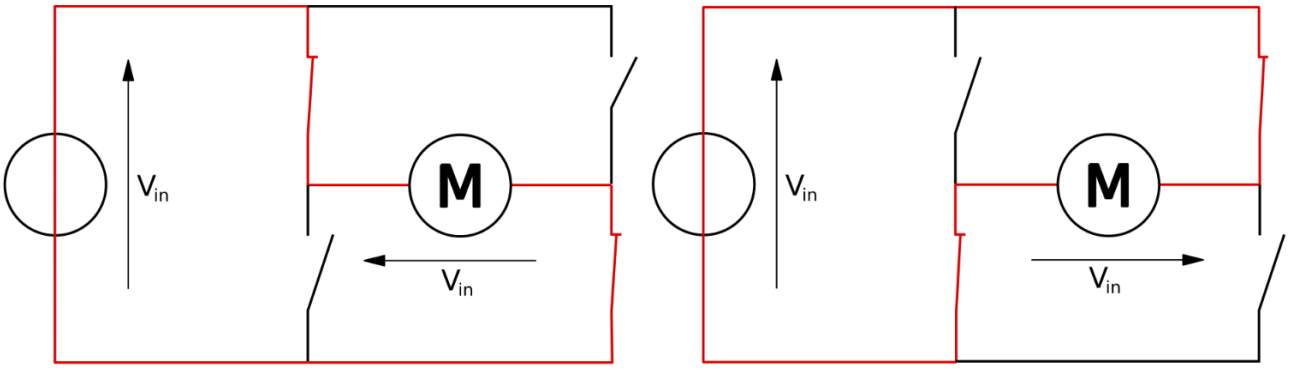
Основні характеристики контролера ESP8266EX[80]:

- підтримка протоколів Wi-Fi 802.11 b/g/n;
- швидкість до 72.2 Мбіт/с;
- дефрагментація;
- два віртуальних інтерфейси Wi-Fi;
- автоматичний пошук мереж;
- підтримка режимів клієнтського пристрою, точки доступу та моніторингу.

5.6 Схема під'єднання двигунів

Двигуни з'єднані з мікроконтролером за схемою H-Bridge. Принцип дії цієї схеми показаний на рисунку 4.3[79]. На рисунку (а) показано увімкнення в додатному напрямку, на рисунку (б) — в протилежному. Мікроконтролер за допомогою транзисторного ключа вмикає та вимикає одночасно два протилежних транзисторних ключі керування двигуном, забезпечуючи обертання двигуна в потрібному напрямку. Керування швидкістю обертання двигуна відбувається за допомогою ШІМ та описано в розділі 61.

Для запобігання перегрівання транзисторів при використанні ШІМ в схему додані діоди зворотного струму (flyback diode)[79].



(а) (б)

Рисунок 4.3 – Принцип дії схеми H-Bridge

6 РОЗРОБКА АЛГОРИТМІВ

6.1 Алгоритми позиціювання

6.1.1 Визначення лінійної швидкости руху

В процесі проєктування системи керування було вирішено використати модуль акселерометра як інерційний модуль для визначення швидкости руху та позиції. Алгоритм визначення позиції за даними акселерометра розроблений на основі опису [80] з модифікаціями.

6.1.1.1 Обробка даних акселерометра

На рисунку 5.1 показано зняті з акселерометра дані щодо прискорення за віссю X, яка вирівняна вздовж повздовжньої осі автомобіля (X). Очевидно, що ці дані вимагають компенсації відносно середнього статичного значення для отримання значущих результатів. X compenstated показує зсунуті відносно цього значення дані.

Після отримання наведених даних необхідно привести їх до стандартних значень (метри на секунду в квадраті) та обробити для усунення механічного шуму. В процесі розробки такого алгоритму виникло питання про кількість значень, які необхідно усереднювати для подальшого інтегрування, оскільки «... важливо отримати середнє значення збалансованої кількості вимірів. Якщо взяти занадто багато вимірів для цього процесу, це може призвести до втрати даних, тоді як взяття занадто малої кількості може призвести до неточного значення»[80, с.4].

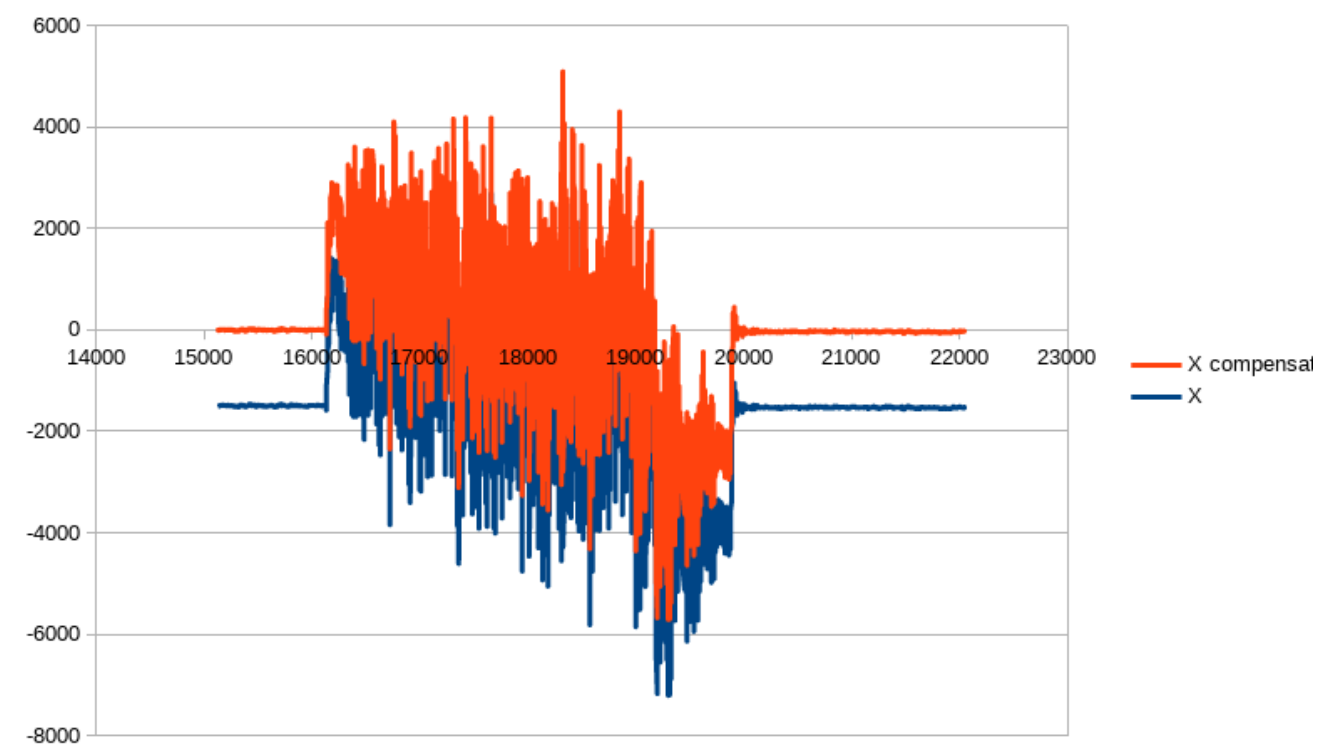


Рисунок 5.1 — Сирі *виміри*, *отримані* з акселерометра

На рисунку 5.2 показано порівняння графіка швидкости (зеленим кольором) та прискорення (жовтим кольором), отриманих при усередненні 80 вимірів, та графіка швидкости, отриманого при інтегруванні значення прискорення без усереднення (блакитним кольором). На рисунку 5.3 наведено аналогічні виміри при усередненні 20 значень.

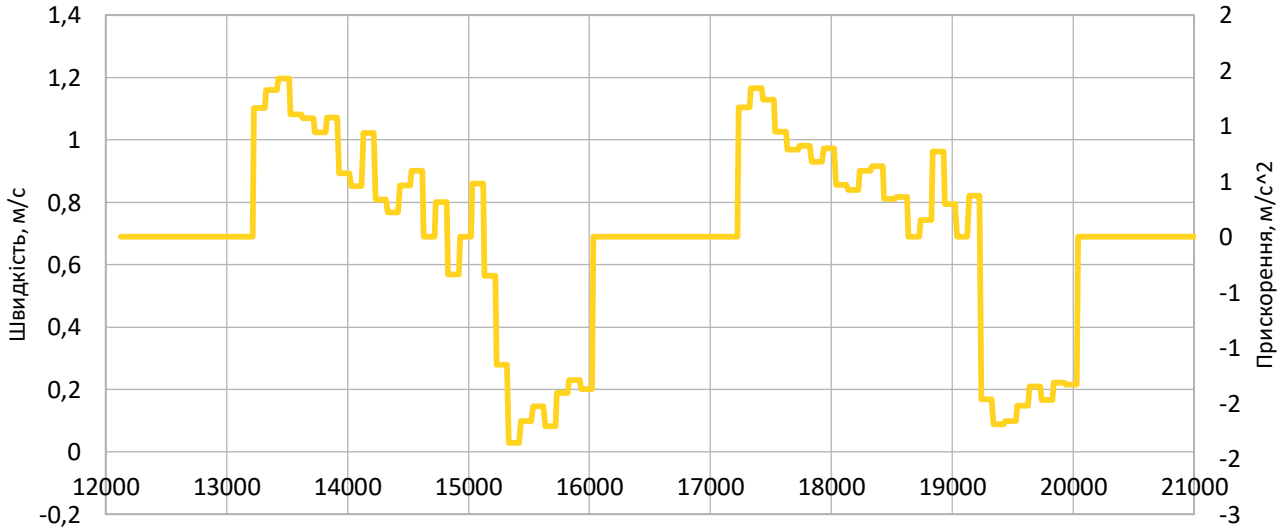


Рисунок 5.2 – Графік швидкости та прискорення при усередненні 80 значень

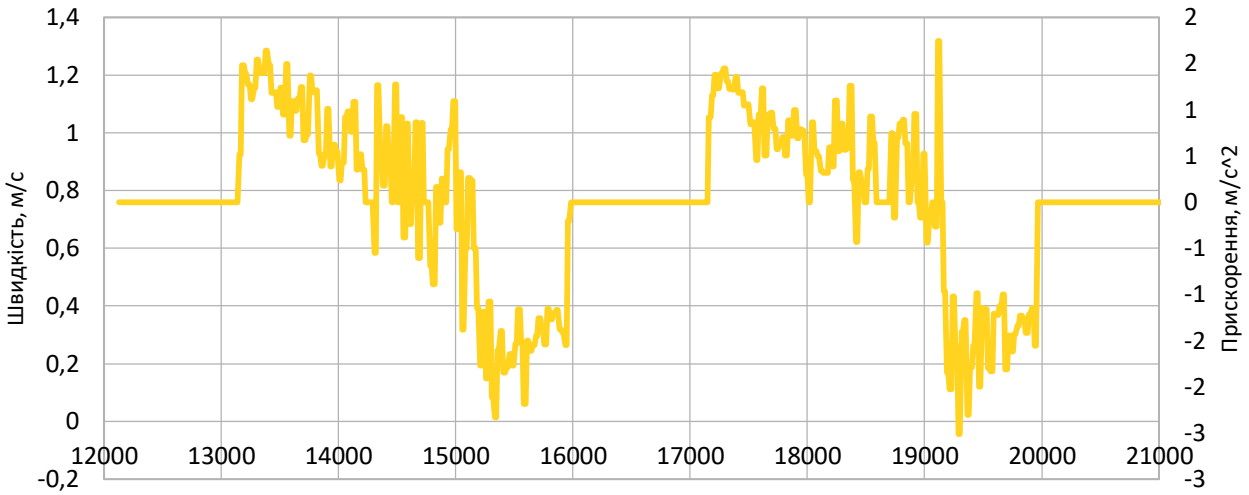


Рисунок 5.3 – Графік швидкости та прискорення при усередненні 20 значень

З наведених рисунків видно, що використання занадто малого усереднення призводить до неточного значення прискорення та, відповідно, швидкости. Тому було обрано усереднення по 80 значень, таким чином, ефективна частота вимірів складає 10 Гц. Окрім того, застосовується алгоритм так званого “discrimination window” для зменшення ефекту механічного шуму в сталому стані — покази акселерометра в сталому стані не повертаються до нуля через особливості структури MEMS-пристрою. Для покращення наступних вимірів також застосовується процедура перекалібрування, згідно з якою при відсутності руху стале значення прискорення, яке використовується для компенсації, оновлюється.

6.1.1.2 Обробка даних швидкости

Отримані усереднені дані прискорення інтегруються шляхом сумування усереднених значень. В статті [80] рекомендується використовувати метод трапецій, однак, при усередненні значень прискорення отримуємо значення на середину, а не початок чи кінець часового проміжку. Таким чином, сума усереднених значень відповідає інтегруванню методом трапецій. Це було практично підтверджено в ході експериментів.

З рисунків 5.2, 5.3 видно, що підсумкове значення швидкости при інтегруванні руху не завжди дорівнює нулю. Для усунення ефекту сталої залишкової швидкості при подальшому інтегруванні для визначення положення, що може призвести до сталого зсуву положення, у випадку повної зупинки пристрою швидкість необхідно скинути до нуля.

Для визначення сталої позиції пристрою, в ході експериментів, найбільш виправданим та ефективним методом виявилась перевірка значень всіх осей акселерометра. У випадку, коли протягом двох часових проміжків усередненння це значення дорівнює нулю, вважається, що пристрій знаходиться в нерухомому режимі. Зразок вимірів прискорення по трьох осях, які підтверджують цю гіпотезу, наведено на рисунку 5.4.

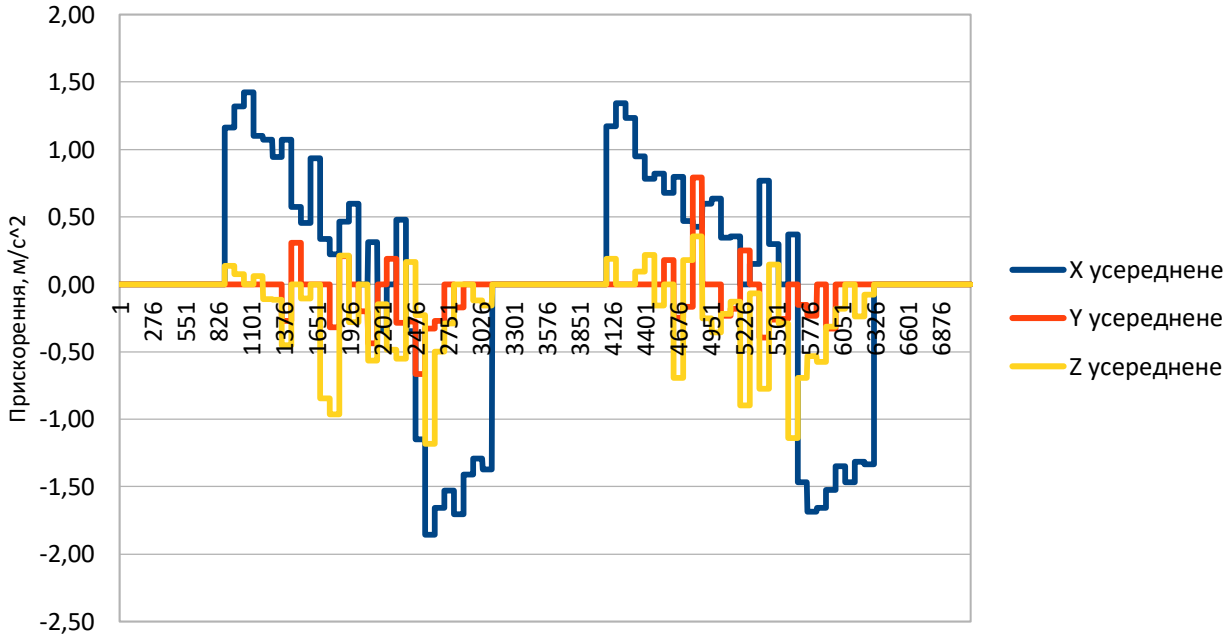


Рисунок 5.4 – Значення прискорення по трьох осях

Після інтегрування виконується повторне чисельне інтегрування даних швидкости для отримання положення пристрою методом трапецій. Це дозволяє зменшити похибку інтегрування. Графік положення пристрою з використанням сирих та фільтрованих даних показано на рисунку 5.5.

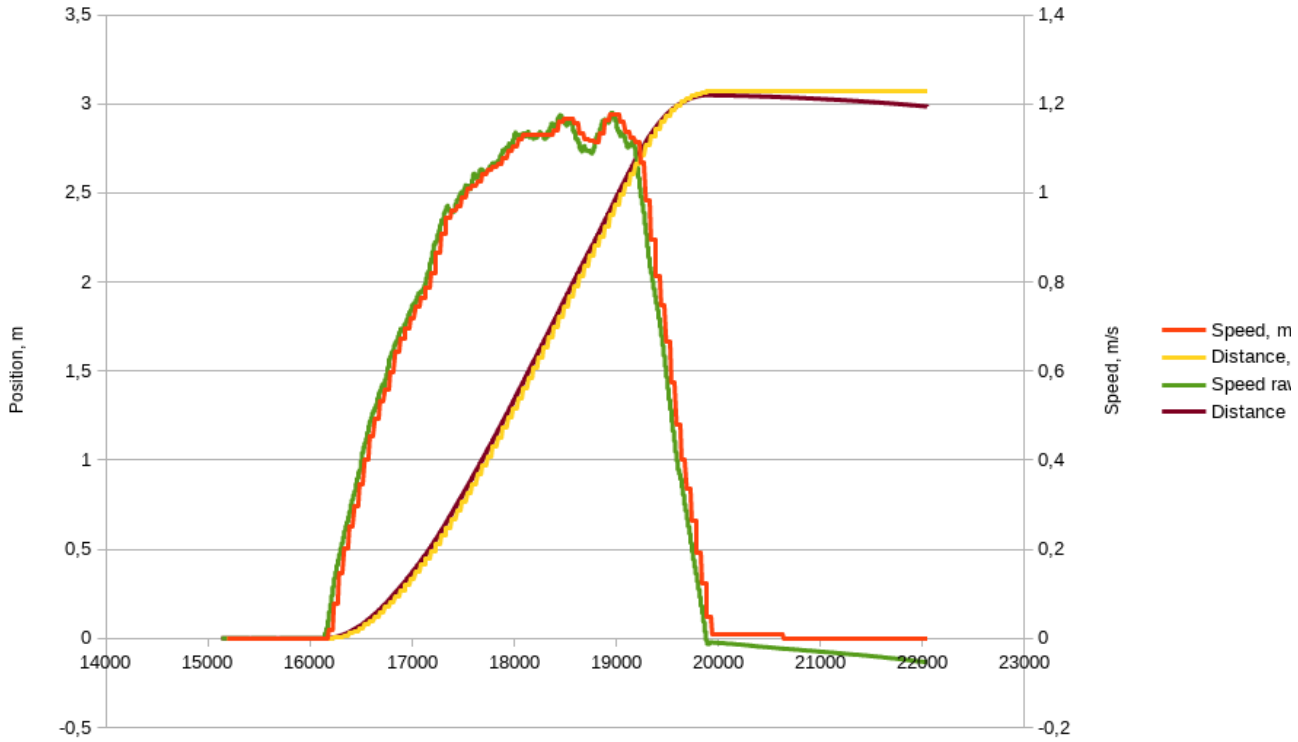


Рисунок 5.5 — Графік швидкости та положення пристрою

Реально виміряний шлях пристрою складає 3.1 метри, тоді як розрахований за даним алгоритмом лежить в межах 3.07 – 3.14 метрів. З цього можна зробити висновок, що даний алгоритм застосовний для визначення положення моделі автомобіля при русі на невеликих відстанях.

6.1.2 Визначення кутової швидкости при повороті

6.1.2.1 Визначення кута повороту за допомогою магнітометра

На початку проєктування було обрано модуль інерційного давача положення з вбудованим магнітометром для визначення кута повороту та кутової швидкости автомобіля при русі. Проте, в процесі проведення досліджень при прямолінійному русі автомобіля було отримано дані магнітометра, показані на рисунках 5.6 – 5.7.

З наведених графіків видно, що вплив електромагнітних завад від двигунів автомобіля неможливо усунути за допомогою алгоритмів обробки даних, що робить використання цього типу давача неможливим для даного пристрою.

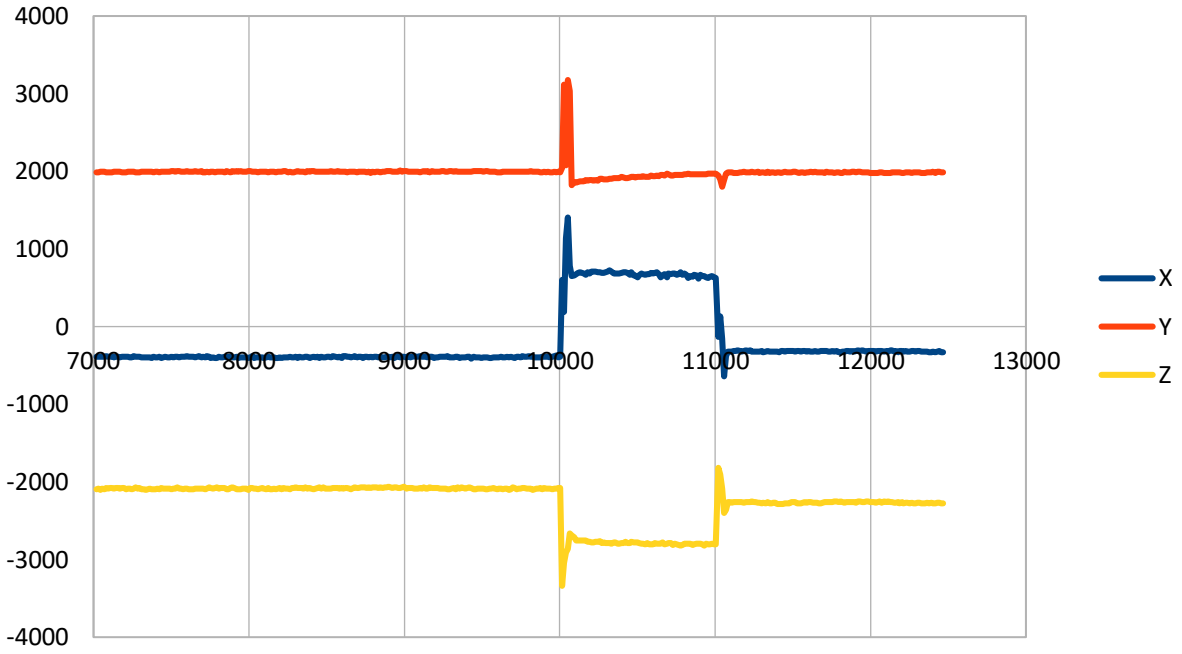


Рисунок 5.6 – Дані магнітометра по осях при русі автомобіля

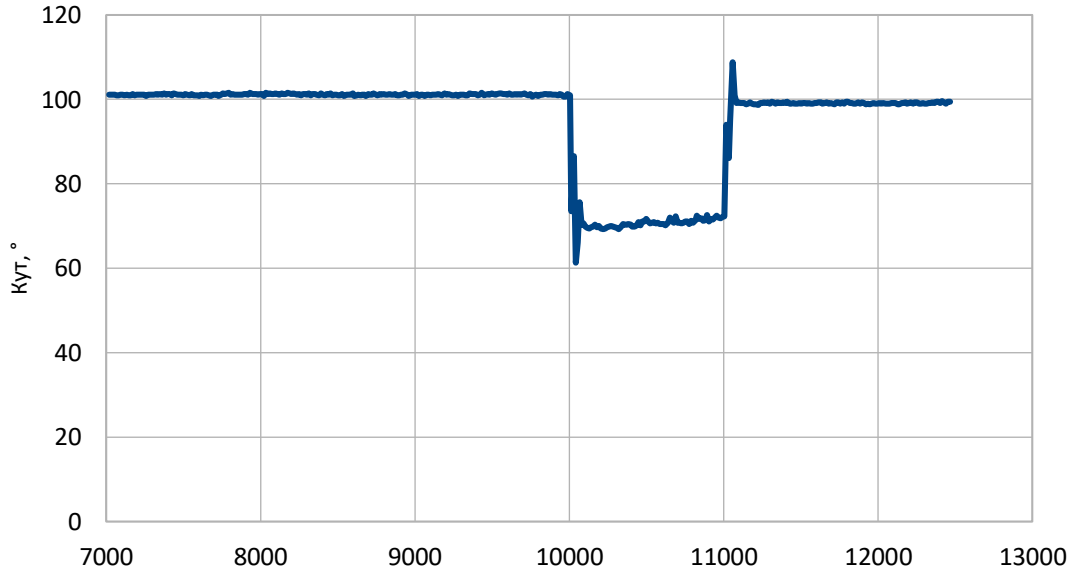


Рисунок 5.7 – Кут, обрахований з даних магнітометра

Для визначення кута повороту та кутової швидкости автомобіля було прийнято рішення використовувати вісь Y даних акселерометра для розрахунку доцентрового прискорення автомобіля, та обчислення кутової швидкості за допомогою

математичних перетворень, а кута повороту — за допомогою інтегрування кутової швидкости методом трапецій.

Спершу було висунуто гіпотезу про еквівалентність прискорення по осі X лінійному прискоренню автомобіля, та прискорення по осі Y доцентровому прискоренню. Таким чином, невідповідністю центру обертання та положення акселерометра було знехтувано. Графік лінійної швидкости, обрахований за цією гіпотезою, показаний на рисунку 5.8. Поворот відбувається з позначки часу 15000 до позначки часу 16000.

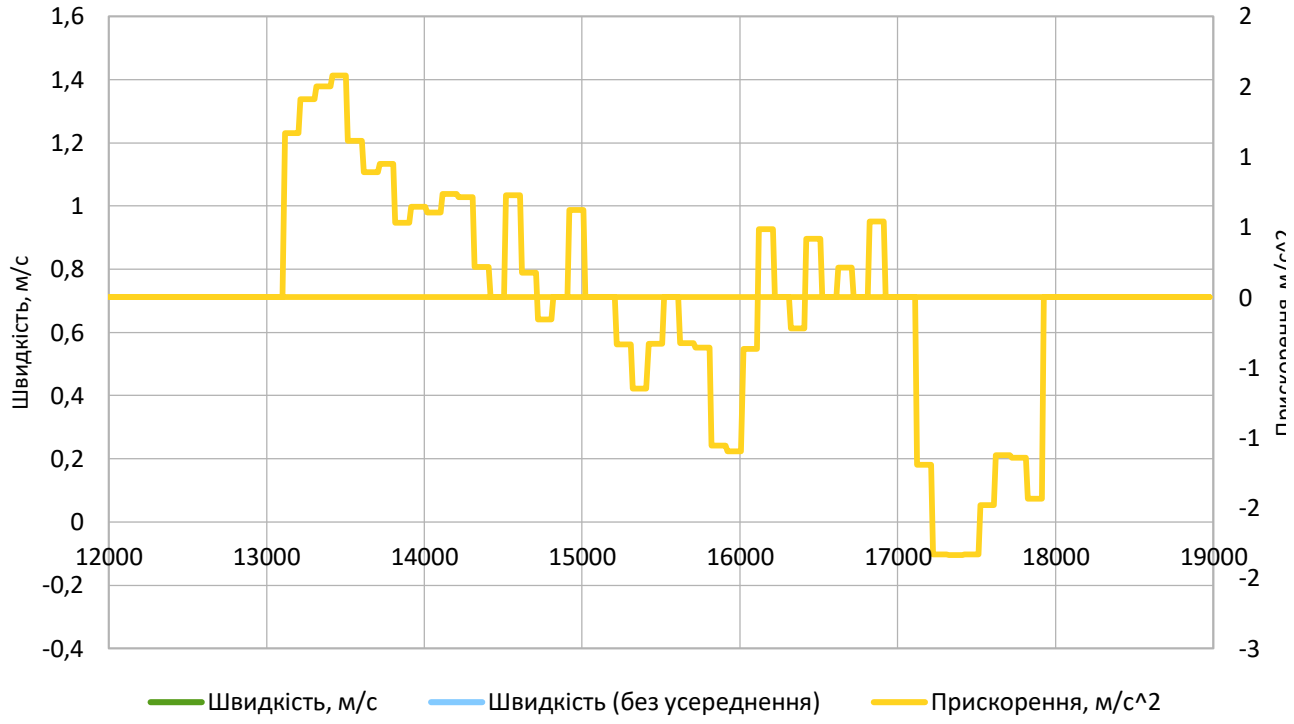


Рисунок 5.8 – Графік швидкости та прискорення при повороті

6.1.2.2 Визначення кутової швидкости з даних прискорення

З наведеного рисунка видно, що нехтування зміщенням осей, за якими діють прискорення, призводить до значної переоцінки від'ємного прискорення, і, відповідно, завеликого зменшення швидкости. Геометричний вигляд прискорень, що

діють на акселерометр при поворотному русі автомобіля, отриманий з математичної моделі автомобіля, описаної в розділі 26, показано на рисунках 5.9 та 5.10.

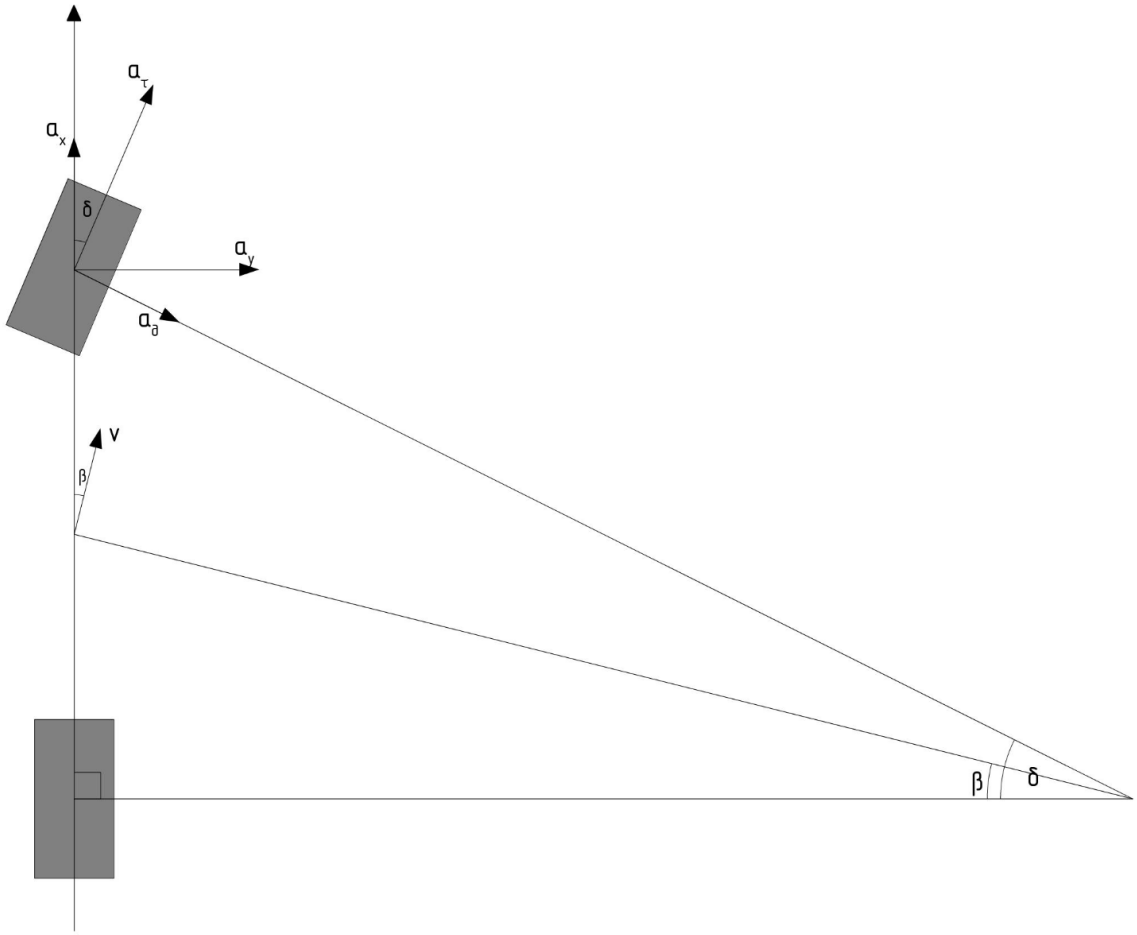


Рисунок 5.9 – Прискорення, що діють на акселерометр, в контексті математичної моделі автомобіля

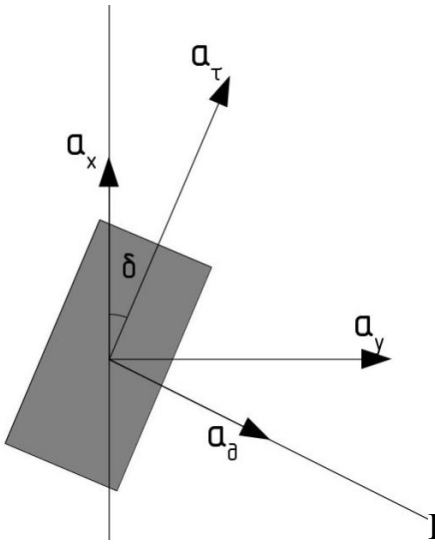


Рисунок 5.10 – Прискорення, що діють на акселерометр (наближено)

Умовні позначення:

$a_д$ — доцентрове прискорення;

a_τ — тангенційне прискорення;

a_x — проєкція прискорень на вісь X;

a_y — проєкція прискорень на вісь Y.

Взаємозв’язок прискорень визначається наступними формулами:

$$\begin{aligned} a_\tau &= a_x\cos(\delta) + a_y\sin(\delta); \\ a_д &= a_y\cos(\delta) + a_x\sin(\delta). \end{aligned} \quad (5.1)$$

Відомо [79, 80], що доцентрове прискорення може бути розраховано з кутової швидкости:

$$a_д = \omega^2 R. \quad (5.2)$$

Також його можна визначити з лінійної швидкости за формулою:

$$\mathrm{a}_д = \frac{v^2}{R}. \quad (5.3)$$

Тоді, кутову швидкість можна визначити через лінійну швидкість та доцентрове прискорення:

$$\omega = \frac{a_д}{v}. \quad (5.4)$$

Для знаходження доцентрового та тангенційного прискорення необхідно визначити кут повороту коліс. Він може відрізнятися залежно від заряду батарей,

швидкости руху автомобіля та інших факторів. Тому необхідно визначити кут повороту з наявних даних.

З формули 3.1 випливає:

$$\frac{\omega l}{v} = \cos(\beta)\tan(\delta).$$

Підставивши β з формули (3.3) та виконавши тригонометричні перетворення, можна отримати наступне рівняння:

$$\frac{\omega l}{v} = \frac{2\tan(\delta)}{\sqrt{4+\tan^2(\delta)}}.$$

Для додатних значень ω та v дійсна тотожність:

$$(\frac{\omega l}{v})^2(4 + \tan^2(\delta)) = 4\tan^2(\delta).$$

Звідси випливає:

$$\delta = \arctan(\frac{2}{\sqrt{(\frac{2v}{\omega l})^2-1}}).$$

Знак кута повороту буде відповідати знаку кутової швидкости, оскільки поворот коліс відбувається в напрямі повороту автомобіля. Тоді остаточний вираз для визначення кута повороту коліс буде виглядати так:

$$\delta = \arctan(\frac{2}{\sqrt{(\frac{2v}{\omega l})^2-1}}) \cdot sign(\omega). \qquad (5.5)$$

Для отримання початкової оцінки кута повороту необхідно отримати початкову оцінку значення кутової швидкости. Такою початковою оцінкою може бути прийняте раніше значення, розраховане з припущення, що доцентрове прискорення дорівнює прискоренню по осі Y. Після розрахунку значення δ з цього припущення, значення a_∂ та a_τ оновлюється, після чого значення δ може бути розраховане більш точно. При використанні послідовного наближення за описаним алгоритмом значення δ збігається, що підтверджено експериментально. Задавши необхідну точність, можемо розрахувати всі параметри системи. Графік розрахованих параметрів наведено на рисунку 5.11.

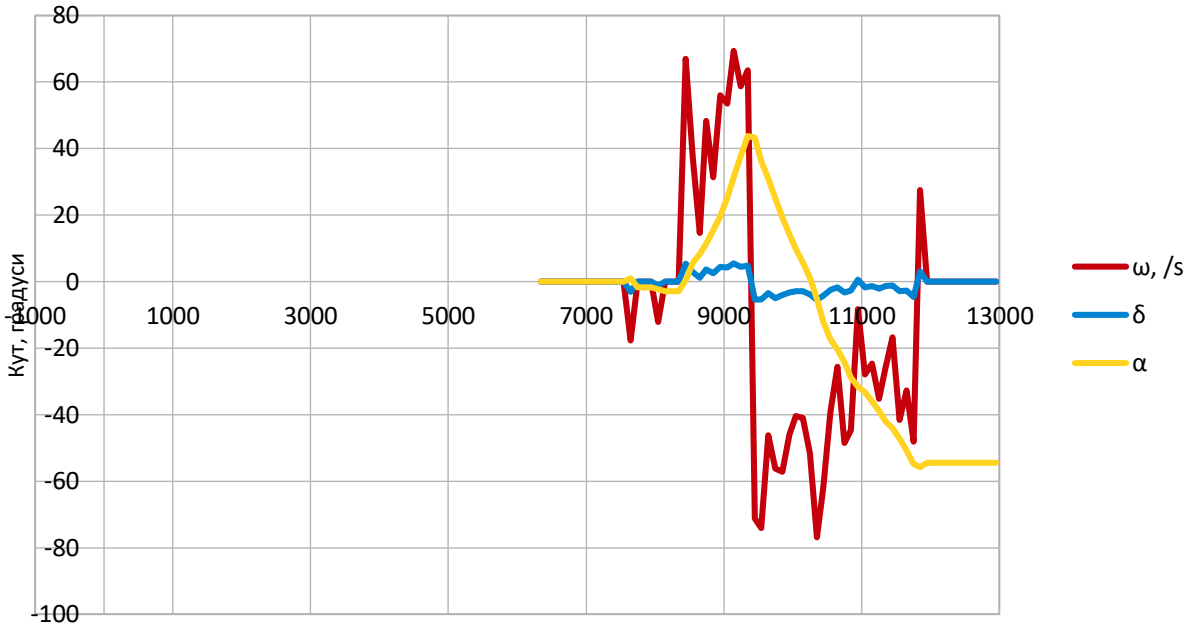
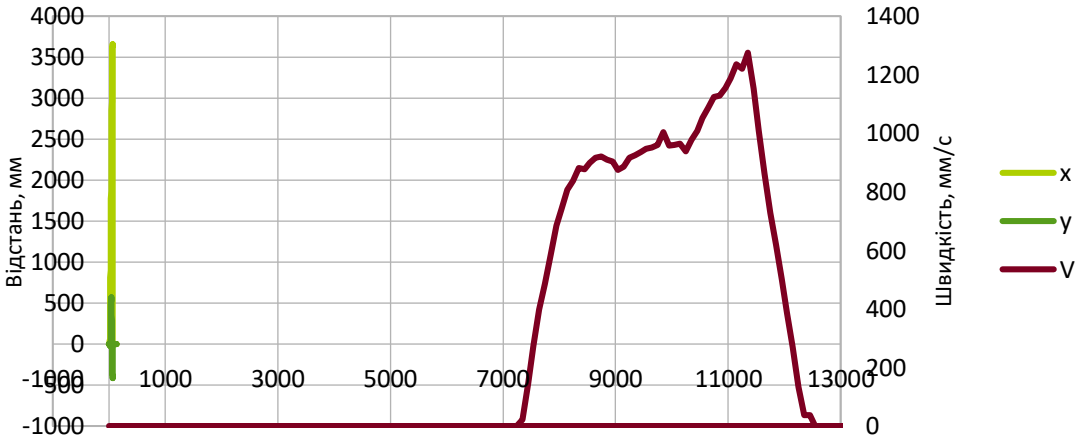


Рисунок 5.11 – Розраховані параметри кутів повороту

Після розрахунку параметрів можемо розрахувати положення автомобіля в проєкціях на осі X та Y за формулами (3.2). Графік швидкости, розрахований з даних тангенційного прискорення, та положення, показаний на рисунку 5.12. Графік руху автомобіля в координатах X та Y показано на рисунку 5.13.



Рисунок

5.12 – Швидкість та положення автомобіля

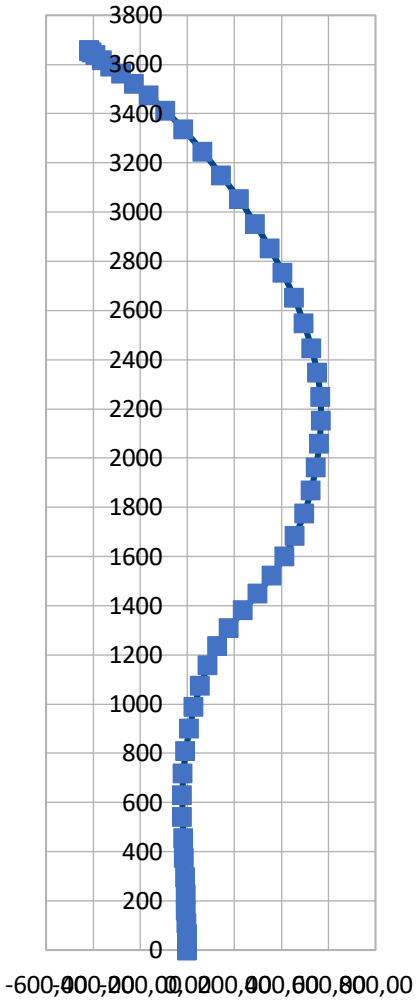


Рисунок 5.13 – *Положення автомобіля в площині XOY*

6.1.3 Опис алгоритму

Блок-схема алгоритму позиціювання наведена в конструкторській документації в кресленику Д4, алгоритму калібрування — на кресленику Д3.

Калібрування виконується наступним чином:

- сума вимірів по кожній осі встановлюється в нуль;
- при кожному перериванні від акселерометра зчитуються дані та додаються до суми по кожній осі, доки кількість підсумованих вимірів менша встановленої кількости вимірів для калібрування;
- сума по кожній осі ділиться на кількість вимірів.

Алгоритм позиціювання викликається при кожному перериванні від акселерометра. Спершу зчитуються дані з внутрішнього буфера акселерометра, після

чого вони компенсуються за допомогою значень калібрування, та отримані прискорення додаються до суми по кожній осі. Якщо кількість підсумованих вимірів достатня для виконання усереднення, виконуються наступні кроки:

- для кожної осі розраховується усереднене прискорення за період як сума вимірів, поділена на їхню кількість;
- якщо усереднене прискорення знаходиться в межах дискримінаційного вікна, воно вважається нульовим;
- якщо прискорення по трьох осях нульове, збільшується кількість нерухомих періодів. Інакше лічильник нерухомих періодів скидається в нуль;
- прискорення по осях X та Y приводиться до міліметрів на секунду;
- кут повороту коліс приймається за нуль;
- виконується послідовне наближення значень тангенційного та доцентрового прискорення. На кожному кроці з отриманих значень розраховується нове значення кутової та лінійної швидкостей, а також кута повороту коліс;
- у випадку, якщо рух не є рухом з поворотом коліс (кут неможливо розрахувати), він приймається за нуль і решта параметрів розраховуються відповідно;
- положення, кут повороту, лінійна та кутова швидкість оновлюється відповідно до розрахованих даних.

Після цього, якщо процедура перекалібрування активна, а пристрій знаходиться в нерухомості принаймні два періоди усереднення, виміряні дані додаються до суми перекалібрування, а якщо кількість вимірів достатня для перекалібрування, розраховується нове значення калібрування по кожній осі як середнє між встановленим та новим значенням. Якщо пристрій рухається, лічильник та суми перекалібрування скидаються в нуль.

6.2 Алгоритм виявлення перешкод

Для позиціювання та сталої роботи алгоритмів побудови маршруту було розроблено алгоритм картографування на основі даних давача відстані на основі алгоритмів, описаних в [79, 79, 80]. Алгоритм використовує дані ультразвукового давача для побудови моделі навколишнього середовища. Ультразвукові виміри обробляються за допомогою імовірнісної моделі давача відстані, поєднуються, акумулюються та зберігаються як матриця імовірности таким чином, щоб звести нанівець неточності та помилкові виміри. Отримані двовимірні карти, або растрові мапи, використовуються алгоритмами планування маршруту для уникнення перешкод та навігації.

6.2.1 Математична модель давача відстані

Імовірнісну модель виміру ультразвукового давача можна визначити наступним чином. Візьмемо отримане з давача значення відстані (вимір часу польоту звуку, помножений на швидкість звуку, приведену до температури середовища та з урахуванням руху пристрою) *R*. З такого виміру можна отримати два висновки:

- на відстані R від давача, в межах ефективного кута вимірювання, знаходиться перешкода;
- на жодній відстані $r < R$, в межах ефективного кута вимірювання, немає перешкоди.

Таку ймовірність ілюструє графік, зображений на рисунку 5.14[79].

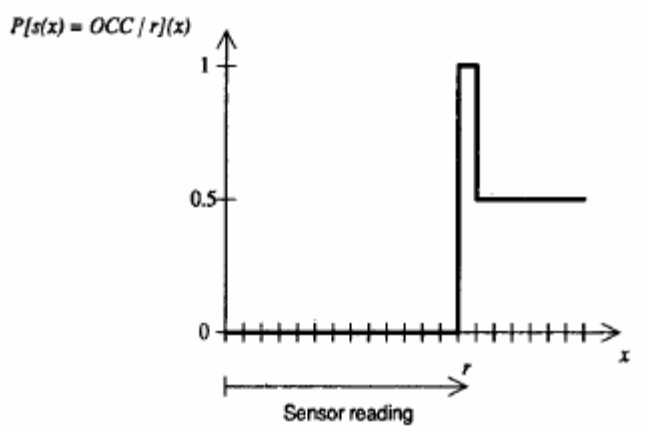


Рисунок 5.14 – *Імовірнісний профіль ідеального давача відстані*

Реальний давач відстані, однак, має більшу імовірність виявити перешкоду, що знаходиться на прямій лінії перед ним (з кутом, близьким до 0°), ніж під певним кутом, а також більшу ймовірність виявити перешкоду, яка знаходиться ближче до нього, ніж перешкоду, що знаходиться далеко. З урахуванням цієї інформації, імовірнісний профіль реального давача відстані зображено на рисунку 5.15[80].

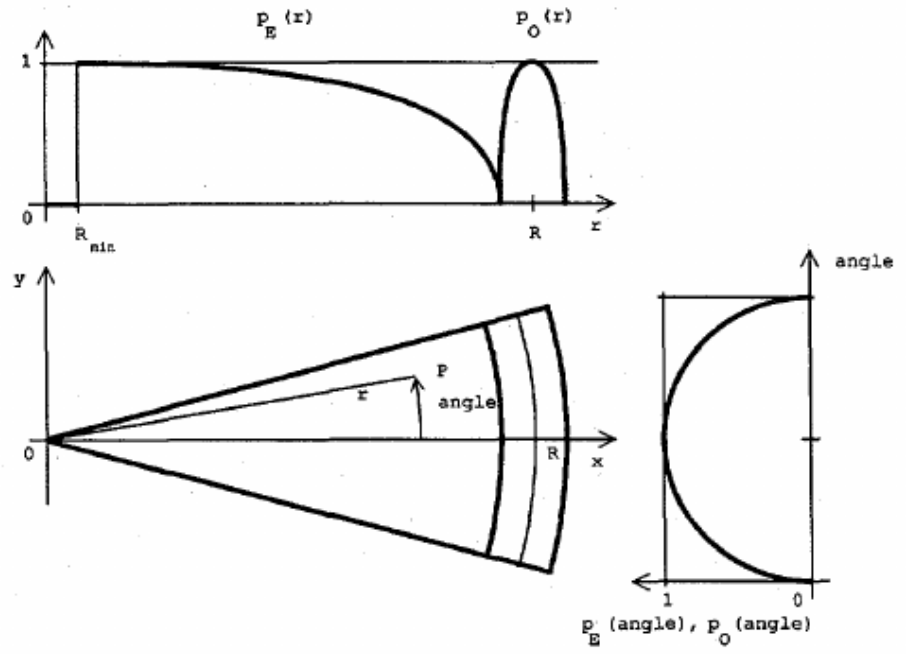


Рисунок 5.15 – *Імовірнісний профіль реального давача відстані*

Тут R — отриманий вимір відстані, а вісь x відповідає осі давача.

Справедлива наступна формула для визначення імовірності того, що комірка з координатами (x, y) порожня, при отриманому вимірі R, для $r(x,y) \in [R_{min}, R)$та $\theta(x,y) \in [-\Omega/2\,, \Omega/2]$:

$$p_E(x,y) = k \cdot E_r(r(x,y)) \cdot E_a(\theta(x,y)) = k \cdot (1 - (\frac{r(x,y)-R_{min}}{R-R_{min}})^2) \cdot (1 - (\frac{\theta(x,y)}{\Omega/2})^2),(5.6)$$

де:

r(x, y) — відстань від точки (x, y) до давача;

θ(x, y) — кут між лінією до точки (x, y) та лінією давача;

R_{min} — мінімальна відстань вимірів;

Ω — ефективний кут вимірів давача;

k — довірчий коефіцієнт виміру.

Оскільки похибка давача не перевищує дискретності растрової мапи[79], імовірність знаходження перешкоди в точці (x, y) при отриманому вимірі R, для точок (x, y), що лежать на дузі *L* (рис. 5.16[79]):

$$p_O(x,y) = \frac{O_a(\theta(x,y))}{\sum_L O_a(\theta)} = \frac{(1-(\frac{\theta(x,y)}{\Omega/2})^2)}{\sum_L O_a(\theta)}. \quad (5.7)$$

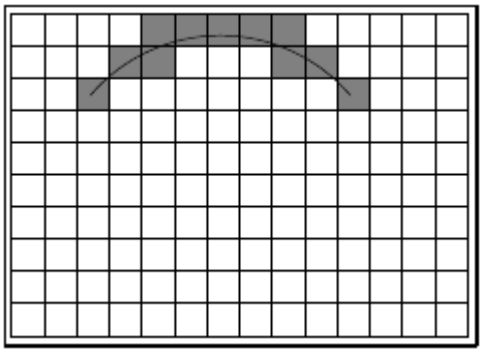


Рисунок 5.16 — Дуга допустимих положень

Розрахунок оновлених імовірностей для кожної точки відбувається за формулою додавання ймовірностей:

$$p(x,y) = p_{r-1}(x,y) + p_r(x,y) - p_{r-1}(x,y) \cdot p_r(x,y), \quad (5.8)$$

за умови$p_E(x,y) = 1 - p_O(x,y)$.

Ефективний кут вимірів давача визначено з документації ультразвукового давача як 30°(рис. 5.18)[80].

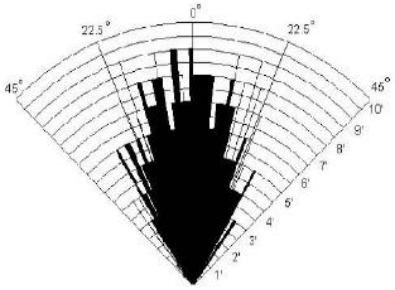


Рисунок 5.17 — Ефективний кут вимірювання

6.2.2 Опис алгоритму

Блок-схема алгоритму наведена в конструкторській документації на кресленику Д6. При кожному оновленні даних від ультразвукового давача розраховується відстань, скомпенсована на швидкість руху. Якщо відстань більша за встановлене верхнє обмеження (3 метри), вважається, що перешкоди не виявлено. Наступним кроком викликається алгоритм оновлення карти перешкод.

Першим кроком алгоритму є визначення області карти, яка оновлюється даним виміром. Для цього будується дуга з центром в давачі відстані, радіусом відповідного виміру та ефективним кутом давача, після чого визначаються її максимальні та мінімальні координати в проєкціях на осі X та Y. Після цього для кожної комірки в даній області виконуються такі кроки:

- виконується перевірка взаємного положення комірки та дуги. Якщо комірка лежить за межами дуги, відбувається перехід до наступної комірки;
- розраховується кут між лінією давача та центром комірки. Якщо кут більший за ефективний кут давача, відбувається перехід до наступної комірки;
- якщо комірка знаходиться всередині дуги, для неї розраховується ймовірність того, що вона порожня, за формулою (5.6), і значення ймовірності в комірці оновлюється за формулою (5.8);
- якщо дуга перетинає комірку, і перешкоду було виявлено, для комірки розраховується ймовірність того, що вона зайнята ($O_a(\theta)$), за чисельником формули (5.7), і ця ймовірність додається до суми ймовірностей на дузі. Після чого

координати комірки та її відповідна ймовірність зберігаються в переліку ймовірно зайнятих комірок.

Після обробки всіх комірок відбувається обробка збереженого переліку ймовірно зайнятих комірок. Для кожної комірки розраховується її ймовірність за формулою (5.7), після чого значення ймовірности перешкоди в комірці оновлюється за формулою (5.8).

Приклад побудованої карти перешкод для двох перешкод показана на рисунку 2.4, для однієї перешкоди — на рисунку 5.18.

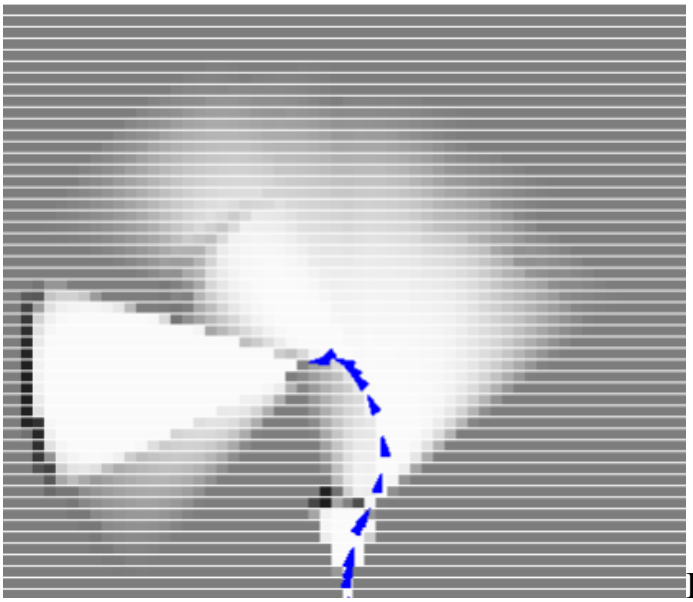


Рисунок 5.18 – Карта перешкод

6.3 Алгоритм планування маршруту

6.3.1 Розробка алгоритму

Алгоритм планування маршруту було розроблено на основі алгоритмів TangentBug[80] та InsertBug[80]. Також враховані деякі моменти, зазначені в алгоритмі WedgeBug[80], зокрема пов’язані з кінематикою робота та обмеженістю поля дії давача відстані.

Алгоритм полягає в чергуванні двох режимів руху робота при русі до цілі. Основний режим називається “рух до цілі” та полягає в побудові маршруту, що

веде до цілі якнайкоротшим шляхом з урахуванням наявних відомостей щодо перешкод. В режимі руху до цілі пристрій рухається по умовній лінії, що з'єднує поточне положення пристрою з положенням цілі, та постійно перевіряє, чи не виявлено перешкоду на цьому шляху. У випадку виявлення перешкоди алгоритм розраховує правий та лівий кінець перешкоди на основі наявних даних та рухається по маршруту, який попередньо передбачає меншу відстань. При початку розвороту автомобіля в напрямку об'їзду перешкоди з'являються нові дані щодо конфігурації перешкод, і минулий шлях уточнюється. Приклад такого руху наведений на рисунку 5.19[80].

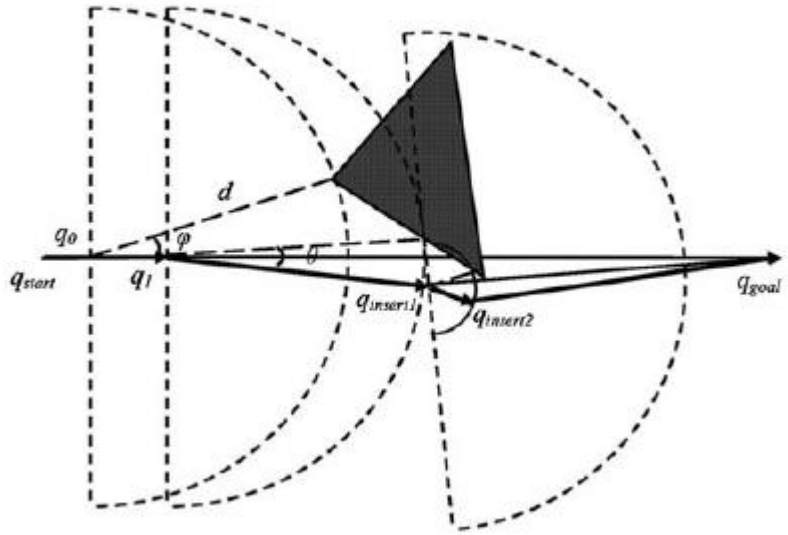


Рисунок 5.19 – Рух в напрямку цілі

Побудований шлях пристрою зберігається у вигляді послідовного набору точок траєкторії запланованого руху. Кожна пара послідовних точок утворює вектор, вздовж якого рухається пристрій. При розгляді можливих напрямків руху з об'їздом перешкоди, вважаються допустимими лише ті вектори, які відхилені від вектора $\overrightarrow{q_0}$(вектор від початкового положення до цілі) не більше, ніж на 90°.

У випадку U-подібної або просто широкої перешкоди може скластися ситуація, що допустимих шляхів не буде знайдено. Це ілюструє рисунок 5.20[80]. Поточна точка позначена літерою P, цільова — літерою T. Обидва вектори до виходу з западини, до точок V_L та V_R, мають кути, більші за 90°.

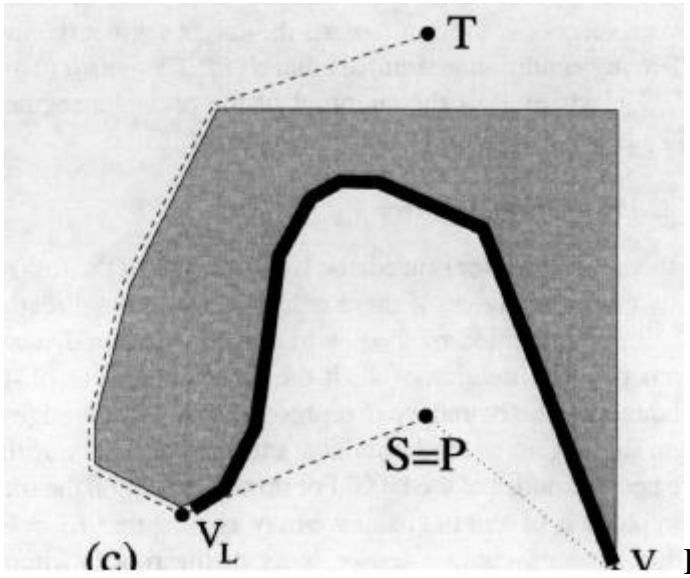


Рисунок 5.20 – *Перешкода з западиною*

У випадку виявлення подібної ситуації алгоритм перемикається в режим розвороту вздовж межі перешкоди. Послідовно розвертаючись в одному напрямку та тримаючи давач відстані направлений в бік перешкоди, пристрій досягне однієї з точок V_L або V_R, після чого шлях до цілі знову може бути побудований. В такому випадку алгоритм знову перемикається в режим руху до цілі. Оскільки карта перешкод збережена, робот не повернеться до попередньої позиції.

6.3.2 Опис алгоритму

Блок-схема алгоритму планування маршруту наведена в конструкторській документації на кресленику Д13. Алгоритм побудови шляху зображений на кресленику Д14.

Після отримання координати цілі, пристрій перемикається в режим руху до цілі. Алгоритм руху описаний в розділі 55. У випадку виявлення перешкоди на шляху, відбувається перепланування маршруту. Рекурсивно запускається функція побудови шляху, яка знаходить координати лівого та правого кінця перешкоди. Після цього запускається побудова шляху до кожного з кінців та від нього до цілі. На кожному кроці обирається коротший зі шляхів оминання перешкоди.

У випадку успішної побудови шляху він зберігається та передається в алгоритм руху для виконання. У випадку неможливості побудувати шлях, запускається

алгоритм розвороту пристрою вздовж перешкоди. Щойно закінчення перешкоди знайдено і рух до цілі може бути відновлено, відновлюється режим руху до цілі.

6.4 Алгоритм руху

Відповідно до алгоритму планування маршруту, маршрут представлений у вигляді векторів між точками маршруту. Таким чином, для досягнення кожної наступної точки необхідно рухатися в бік її знаходження. Це здійснюється за допомогою підтримання сталої швидкости руху (зменшується з урахуванням відстані до перешкод) та кута напрямку руху, який має збігатися з кутом поточного вектора маршруту.

6.4.1 Регулювання швидкости руху

Для регулювання швидкости руху був застосований пропорційно-інтегральний регулятор. Схема регулятора зображена на рисунку 5.21[80].

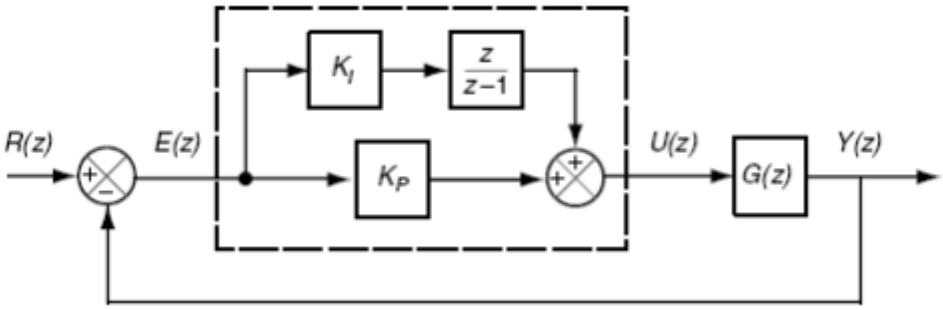


Рисунок 5.21 – Схема ПІ–регулятора

Порівняння розгінних характеристик системи з пропорційним та пропорційно-інтегральним регулятором швидкости наведене на рисунку 5.22. З графіка видно, що застосування пропорційного регулятора дає сталу похибку, яка може бути усунута використанням інтегрувальної ланки. Також видно, що швидкість нестабільна, що свідчить про неможливість використання диференціювальної ланки.

В проєкті застосовано наступні коефіцієнти ПІД-регулятора:

$K_i=0.4;$

$K_p=0.003;$

$K_d=0$.

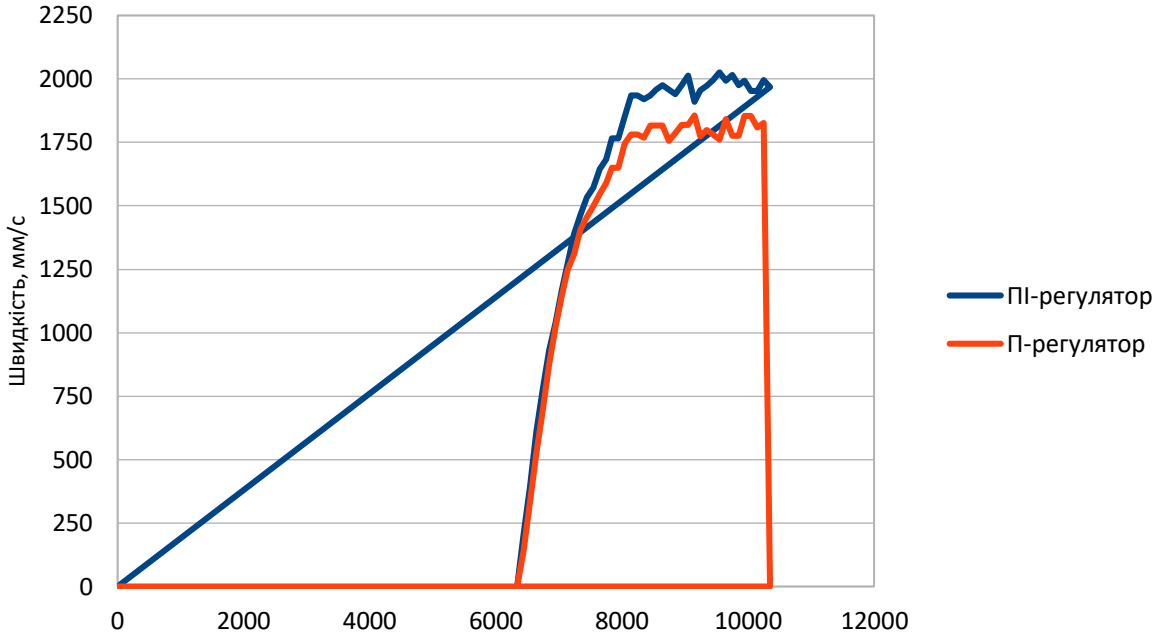


Рисунок 5.22 – Розгінна характеристика

6.4.2 Опис алгоритму

Блок-схема алгоритму руху за маршрутом наведена в конструкторській документації на кресленику Д12. Відповідно до алгоритму планування маршруту, є два режими руху пристрою — рух до цілі та слідування вздовж межі. В першому режимі алгоритм руху виглядає наступним чином.

З кожним оновленням даних положення відбувається перерахунок швидкості руху. Якщо перешкода або ціль знаходяться на відстані, ближчій, ніж шлях гальмування, швидкість зменшується пропорційно до відстані до перешкоди або цілі, а якщо знаходиться ближче, ніж радіус безпеки — виконується зупинка. Після цього розраховується регулювальний вплив за допомогою ПІ-регулятора та встановлюється відповідна потужність двигуна.

Розраховується кут між напрямком до найближчої точки маршруту та напрямком руху автомобіля. Якщо відхилення значне, вмикається поворот передніх коліс для зміни кута руху.

В другому режимі, з урахуванням кінематики та механічної конфігурації автомобіля, рух вздовж перешкоди вперед не є можливим. Натомість виконуються поступальні рухи розвороту вперед-назад, тримаючи давач відстані напрямленим в бік перешкоди. При русі назад неможливо визначити безпечність такого руху, тому рух здійснюється мінімальний період часу та лише по “достовірно вільній” частині карти перешкод.

7 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

Загальна структура системи у вигляді діаграми розгортання показана на рисунку 6.1.

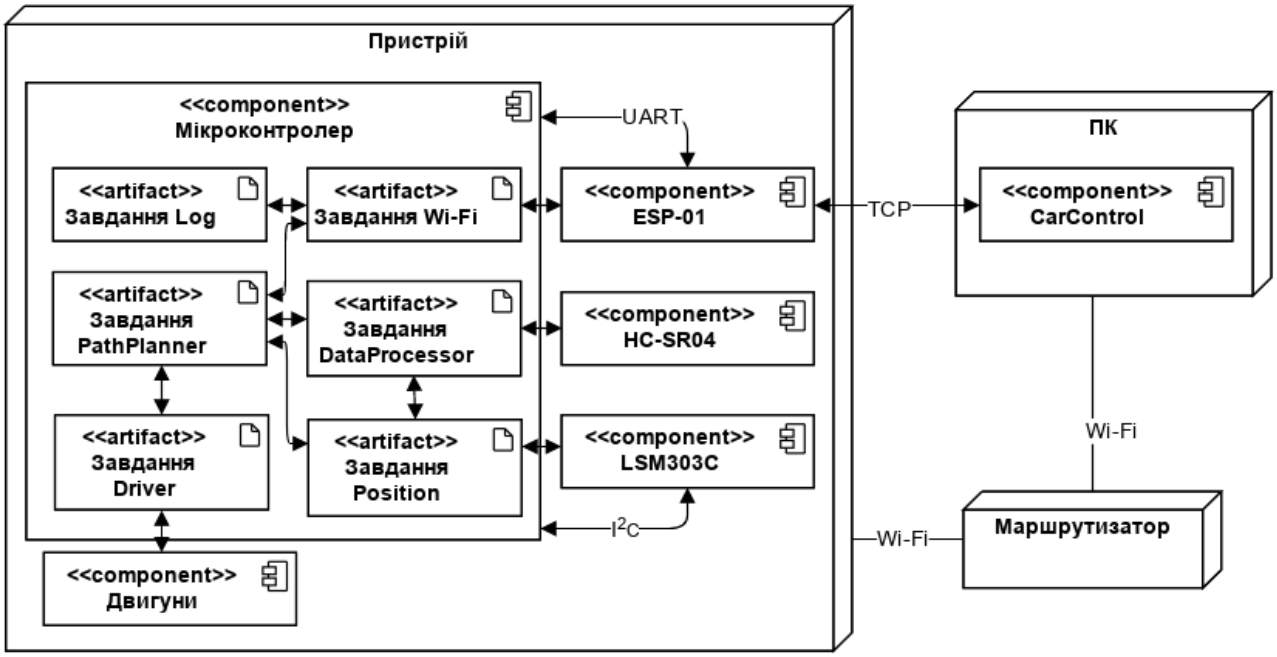


Рисунок 6.1 – Діаграма розгортання системи

Програмний проєкт складається з двох частин, системи керування моделлю автомобіля та системи візуалізації телеметричних даних. Система візуалізації виконується на ПК, система керування — на мікроконтролері. З'єднання між частинами системи відбувається за допомогою протоколу Wi-Fi.

7.1 Система керування автомобілем

7.1.1 Структура системи керування

Структура системи керування наведена на рисунку 6.2 та в кресленику Д1 конструкторської документації. Програма написана мовою C++. Код проєкту доступний за посиланням в [81]. Діаграма класів наведена в конструкторській документації в кресленику Д17.

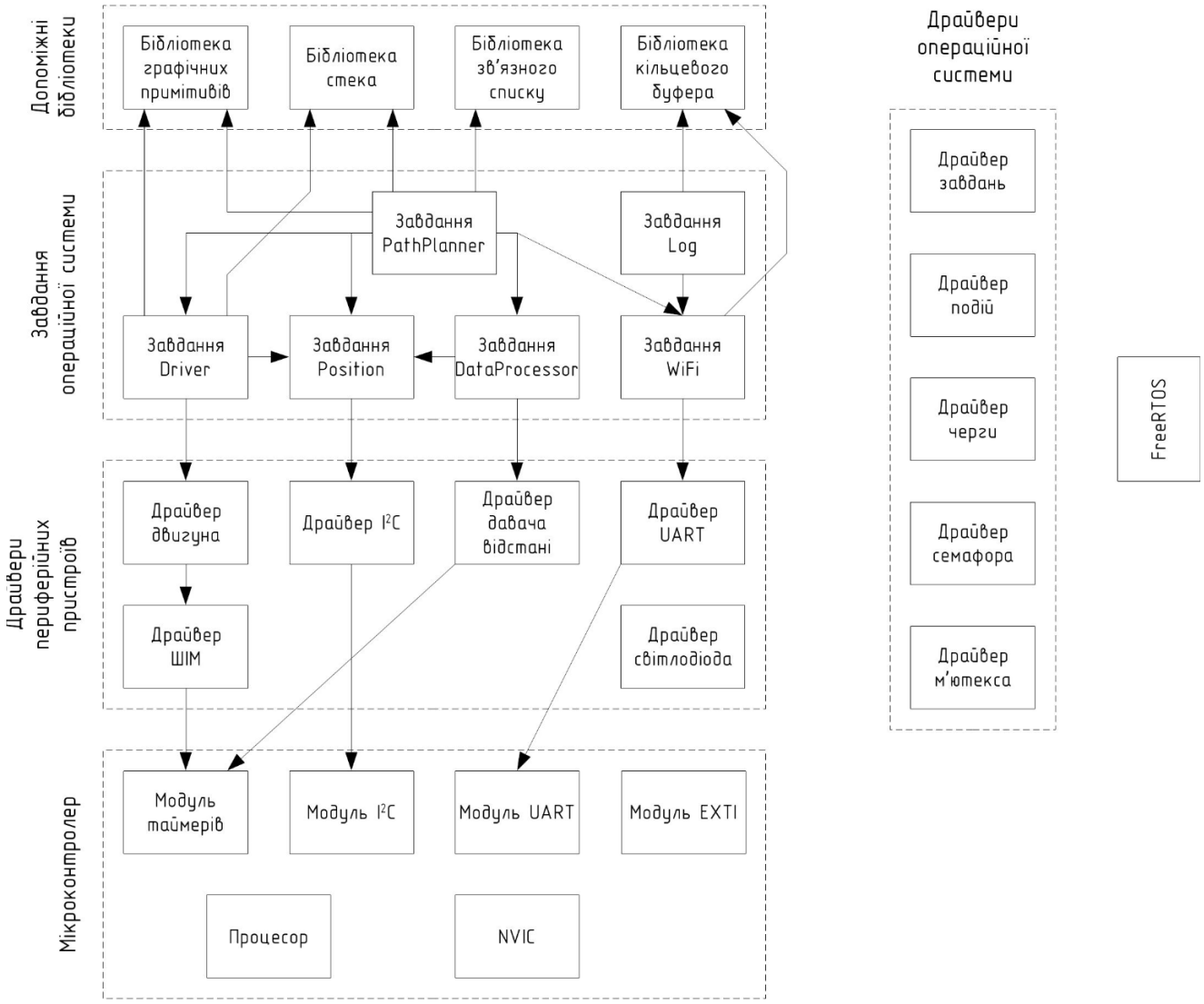


Рисунок 6.2 – Структура програмної системи

Елементи структури:

- мікроконтролер включає процесор, контролер переривань та периферійні модулі. На схемі наведено найважливіші модулі з тих, які використовуються в програмі;

- драйвери периферійних пристроїв — розроблені драйвери для роботи з периферійними модулями мікроконтролера. Зокрема, розроблені драйвери для протоколу I^2C, світлодіода, широтно-імпульсної модуляції, контролю двигунів за допомогою ШІМ, зв'язку за протоколом UART (УАПП), та ультразвукового давача відстані.

- драйвери ОС — обгортки примітивів синхронізації ОС для потреб розроблюваної системи мовою C++. Розроблені драйвери синхронізації за допомогою подій (eventgroups), м’ютексів, черг та семафорів, а також драйвер-обгортка завдань ОС. Залежності від драйверів ОС не показано на схемі.

- допоміжні бібліотеки — розроблений клас алгоритмів для роботи з кільцевим буфером (зберігання та обробки отриманих даних або даних, що знаходяться в черзі надсилання), стеком (зберігання точок шляху), зв’язного списку (для побудови шляху);

- завдання операційної системи — розроблені завдання операційної системи, що працюють протягом всієї роботи пристрою та виконують певні алгоритми. Їх можна поділити на три класи:

- низькорівневі завдання — завдання, що використовуються для зв’язку та першочергової обробки даних під’єднаних пристроїв і не залежать від інших завдань. До цього класу належать завдання WiFi (взаємодія з модулем Wi-Fi ESP-01) та Position (взаємодія з модулем акселерометра та магнітометра LSM303C);

- завдання середнього рівня — завдання, що взаємодіють із під’єднаними пристроями з використанням даних, отриманих від інших завдань. До цього класу належать завдання Driver (керування двигунами з використанням даних положення, отриманих від завдання Position, а також даних шляху, отриманих від завдання PathPlanner) та DataProcessor (обробка даних давача відстані з використанням даних положення);

- високорівневі завдання — завдання, що не взаємодіють з пристроями, а лише обробляють дані та керують поведінкою інших завдань. До цього класу належать завдання PathPlanner (побудова маршруту на основі даних, отриманих від завдань Position та DataProcessor, та керування роботою завдання Driver) та Log (збір даних від усіх завдань та почергове надсилання інформації за допомогою завдання WiFi).

Для роботи програми використовуються також наступні сторонні бібліотеки та засоби:

- FreeRTOS — операційна система реального часу [80];

- HAL (Hardware Abstraction Layer) — бібліотека файлів з визначеннями адрес регістрів мікроконтролера та функцій доступу до регістрів, що постачається компанією ST Microelectronics [81];

- CMSIS (Cortex Microcontroller Software Interface Standard) — бібліотека підтримки мікропроцесора Cortex-M4, що постачається компанією ARM.

7.1.2 Драйвери периферійних пристроїв

7.1.2.1 Драйвер ШІМ

Широтно-імпульсна модуляція використовується для керування середнім значенням вихідної напруги шляхом зміни тривалости імпульсів у цифровій системі [79]. В мікроконтролерах STM вона виконується за допомогою таймерів. Приклад роботи таймера для створення одного імпульсу ШІМ наведено на рисунку 6.3[81].

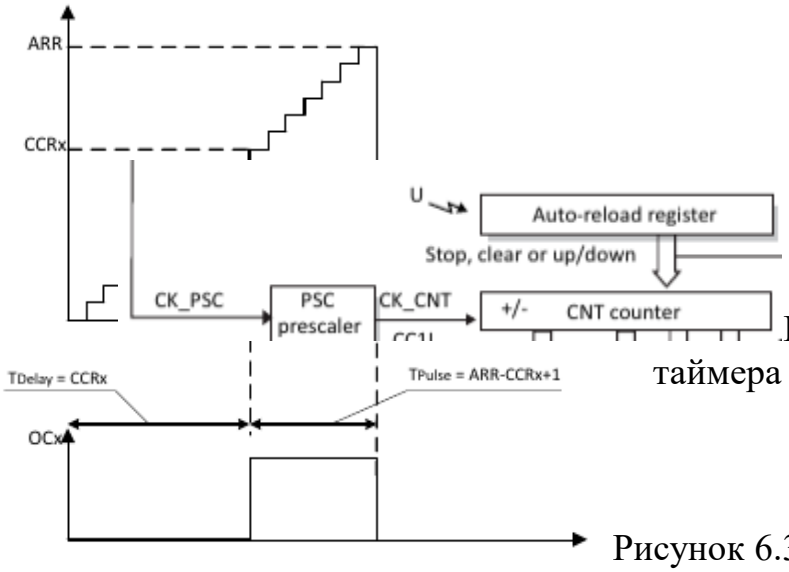


Рисунок 6.3 – Імпульс ШІМ

Рисунок 6.4 – Модуль підрахунку таймера

На рисунку 6.4[81] зображено модуль підрахунку таймера. На подільник частоти подається тактовий сигнал із частотою шини, до якої під'єднаний таймер. Частота широтно-імпульсної модуляції визначається за формулою:

$$f_{PWM} = \frac{f_{cnt}}{ARR+1} = \frac{f_{ck}}{(PSC+1)\cdot(ARR+1)}. \quad (6.1)$$

На рисунку 6.5[81] зображено блок керування виводом таймера. Генерація ШІМ відбувається за допомогою перемикання вихідного сигналу, коли значення рахунку (CNT/ counter) та регістра порівняння (CCR/ Capture/Compare Register) збігаються. Для регулювання ШІМ в межах від 0 до 100% значення ARR встановлюється в 99, тоді при значенні CCR в 100 відбувається повне заповнення ШІМ, а при значенні 0 — нульовий сигнал. Значення подільника частоти розраховується з заданого значення частоти та частоти шини.

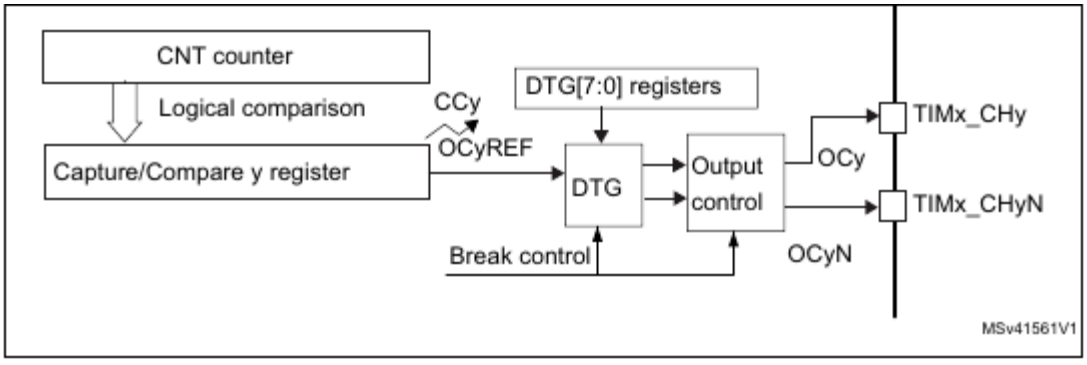


Рисунок 6.5 – Блок керування виводом таймера

7.1.2.2 Драйвер двигуна

Двигуни під'єднано до мікроконтролера з використанням схеми включення типу Н-міст, описаної в розділі 3З.

Для роботи з двигуном використовується два канали ШІМ таймера. При русі вперед на позитивний полюс двигуна подається ШІМ із заданим коефіцієнтом заповнення, а на негативний контакт — нуль. При русі назад на негативний полюс двигуна подається ШІМ, а на позитивний — нуль.

Драйвер забезпечує атомарність увімкнення двигуна, таким чином, уникаючи ситуації подачі напруги на обидва транзисторних ключі, що може призвести до короткого замикання

7.1.2.3 Драйвер протоколу I²C

I²C — послідовний протокол обміну даними між мікроконтролерами, що використовує лише два провідники та дозволяє під'єднувати кілька пристроїв на одну шину. В цьому його перевага над такими протоколами, як UART та SPI.

Приклад послідовності передавання даних наведено на рисунку 6.6.

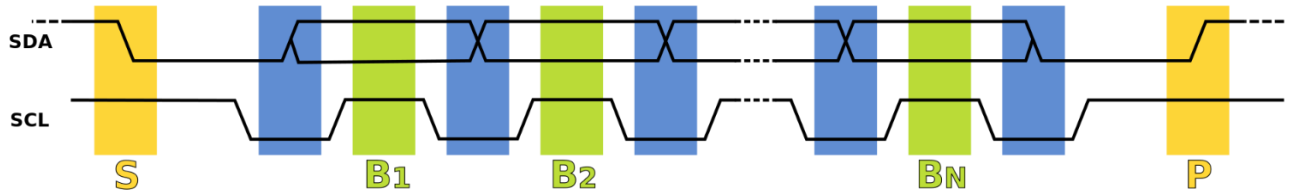


Рисунок 6.6 – Послідовність передавання даних по шині I²C

На початку взаємодії керівний пристрій подає сигнал START, після чого передає адресу периферійного пристрою. Якщо пристрій з такою адресою під'єднаний до шини, він відповідає сигналом ACK. Після цього на пристрій передається адреса внутрішнього регістра, а потім, у випадку запису даних — послідовно передаються байти даних, а у випадку зчитування виконується операція ReStart. Після операції ReStart передається адреса пристрою з позначкою "читання" та починається отримання даних. Після отримання даних виконується операція Stop.

На рисунку 6.7 показана діаграма станів драйвера протоколу I²C (також наявна в конструкторській документації на кресленику Д15). В даному драйвері надсилання даних виконується за допомогою переривань (оскільки такий запис здійснюється лише один раз на початку роботи програми), а отримання даних — за допомогою контролера прямого доступу до пам'яті (DMA – Direct Memory Access Controller), що дозволяє звільнити процесорний ресурс на час отримання даних.

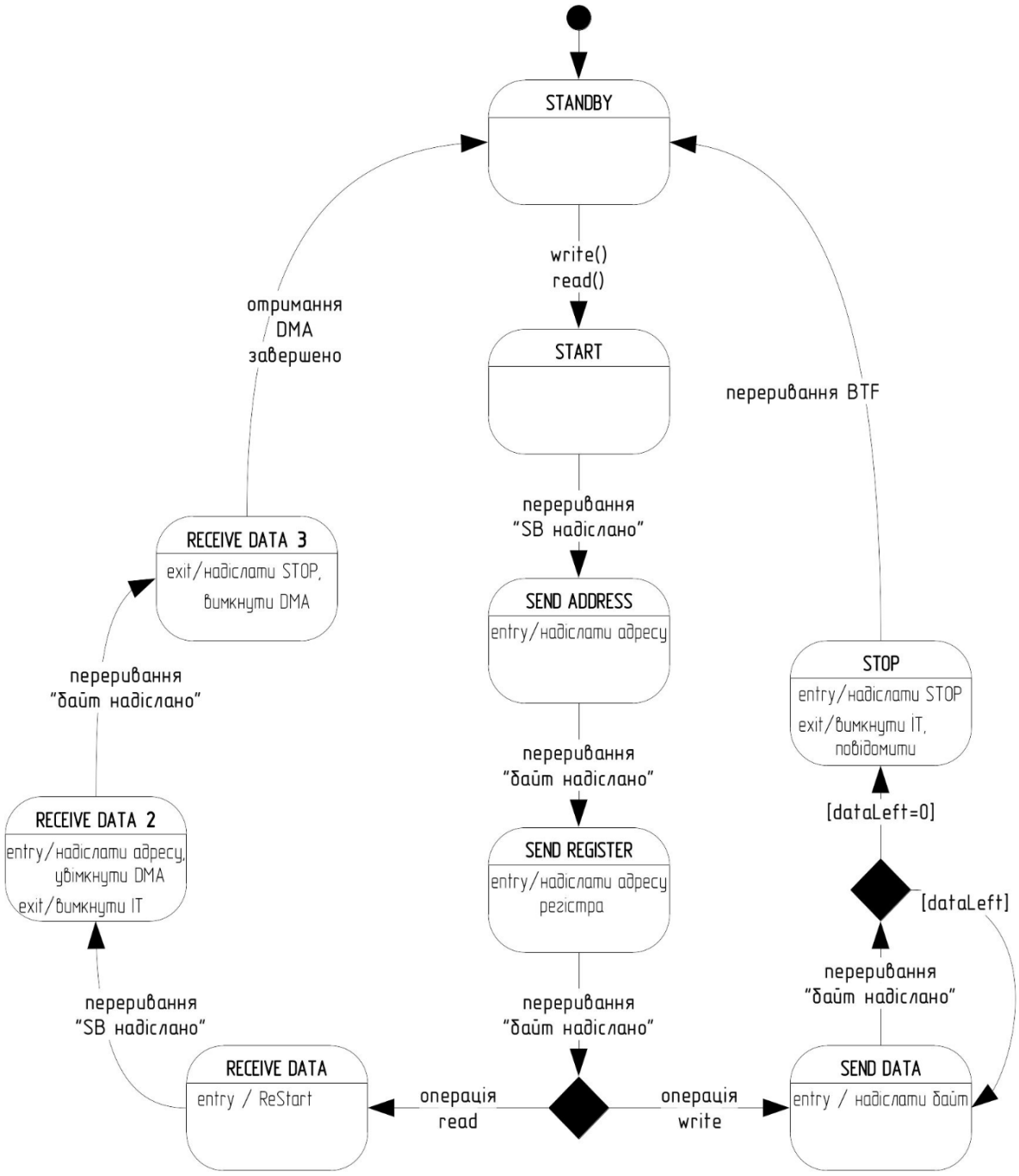


Рисунок 6.7 – Діаграма станів драйвера I^2C

7.1.2.4 Драйвер давача відстані

Відповідно до протоколу роботи з давачем відстані, описаному в розділі 31, для надсилання ультразвукового пакета необхідно подати імпульс тривалістю 10 мікросекунд на лінію Trig, та виміряти тривалість сигналу на лінії Echo. Для надсилання сигналу було застосовано таймер в режимі ШІМ аналогічно до описаного

в розділі 61, із заповненням 1/10000 на частоті 10 Гц. Таким чином, імпульс тривалістю 10 мікросекунд подається на лінію Trig кожні 100 мілісекунд. За потреби, якщо відбитий сигнал було отримано раніше, можливе перезавантаження таймера для надсилання нового імпульсу.

Вимірювання тривалости польоту звуку також відбувається за допомогою таймера. Структура блоку захоплення вхідного сигналу таймера мікроконтролера STM32 показана на рисунку 6.8. Після програмування захоплення заднього фронту сигналу, при виявленні відповідного фронту генерується подія захоплення (capture event), і в регістр захоплення (CCR / Capture/Compare Register) зберігається поточне значення лічильника (CNT – Counter).

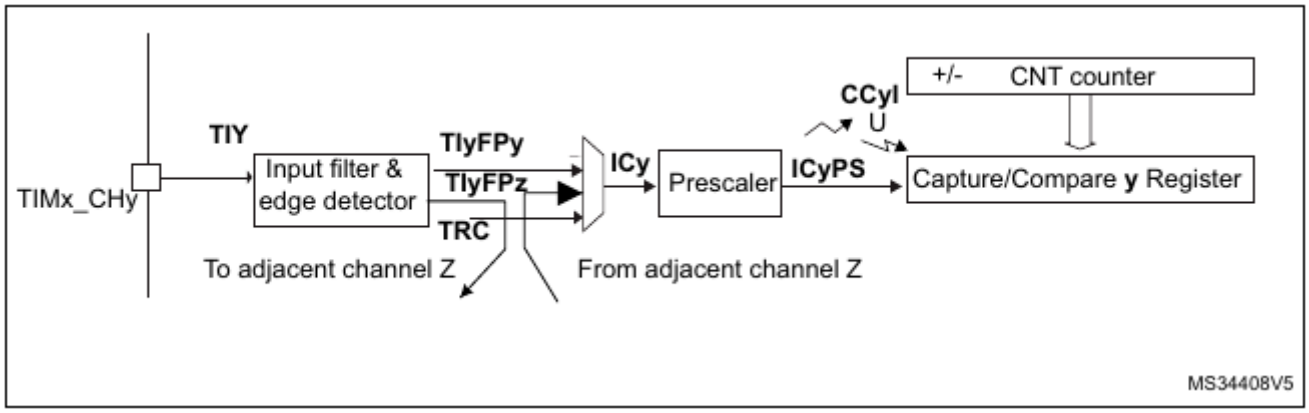


Рисунок 6.8 – Блок захоплення вхідного сигналу таймера

Частота лічильника таймера визначаться за формулою:

$$f_{cnt} = \frac{f_{ck}}{PSC+1}. \quad (6.2)$$

В даному драйвері частоту подільника розраховано таким чином, щоб частота лічильника дорівнювала 1 МГц, таким чином дозволяючи вести відлік в мікросекундах. Для захоплення початку сигналу використано режим перезапуску від зовнішнього тригера, де як тригер використано передній фронт сигналу, як показано на рисунку 6.9[81].

Після отримання переривання драйвер зберігає виміряне значення часу. При виклику функції отримання відстані, розраховується відстань за формулою:

$$R = (v_{\text{звуку}} - v_{\text{руху}}) \cdot \frac{\tau}{2} [\text{мм}]. \quad (6.3)$$

Швидкість звуку розраховується з формули [79]:

$$v_{\text{звуку}} = 331300 + 596 \cdot t°[\text{мм/с}]. \quad (6.4)$$

Температура навколишнього середовища встановлюється програмою з даних давача температури в модулі магнітометра.

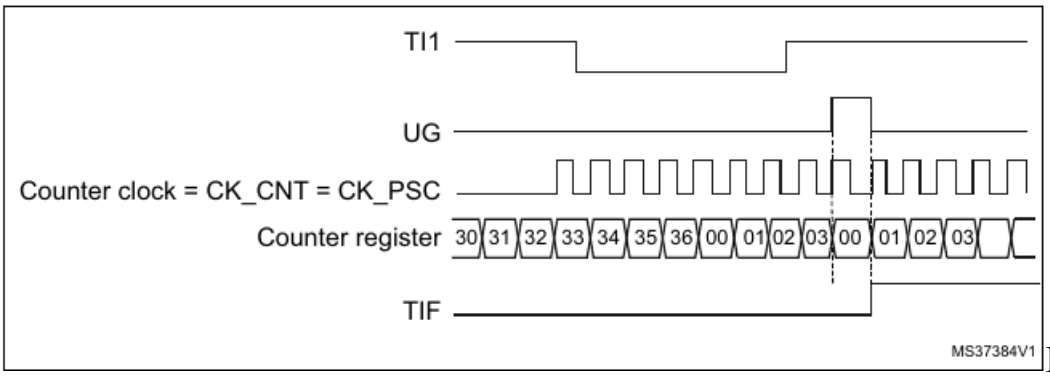


Рисунок 6.9 – Режим перезав*антаження* від зовнішнього тригера

7.1.2.5 Драйвер UART

UART (Universal Asynchronous Receiver/Transmitter), або УАПП (універсальний асинхронний приймач-передавач) — протокол асинхронного послідовного передавання даних з використанням двох ліній, однієї для передавання даних в одному напрямку та іншої — в протилежному. Приклад передавання байта даних за допомогою УАПП наведено на рисунку 6.10[79].

Рисунок 6.10 – Передавання байта даних за допомогою УАПП

Протокол УАПП застосовується для обміну даними між двома рівноправними мікроконтролерами та передбачає попереднє узгодження параметрів передавання, таких як частота, довжина байта, кількість стоп-бітів, тип біта парности та інші. В даному проєкті реалізований обмін даними на частоті 1 МГц з довжиною байта 8 бітів.

Використання модуля UART мікроконтролера STM32 для приймання та передавання даних в блоковому режимі з використанням DMA передбачає невизна-

ченість щодо довжини вхідних даних. В цьому випадку можна використати кільцевий режим роботи DMA та обробляти вхідні дані за перериваннями “половина буфера отримана”, “буфер заповнений” та “лінія UART неактивна”. Останній випадок дозволяє визначити завершення передавання блоку даних, коли таке передавання не збігається з кінцем буфера отримання. Приклад такого передавання показаний на рисунку 6.11[81].

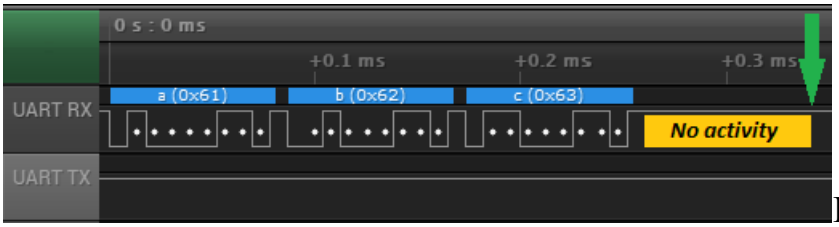


Рисунок 6.11 – Передавання блоку даних

Переривання, що генеруються модулями UART та DMA при отриманні даних в режимі кільцевого буфера показано на рисунку 6.12[81].

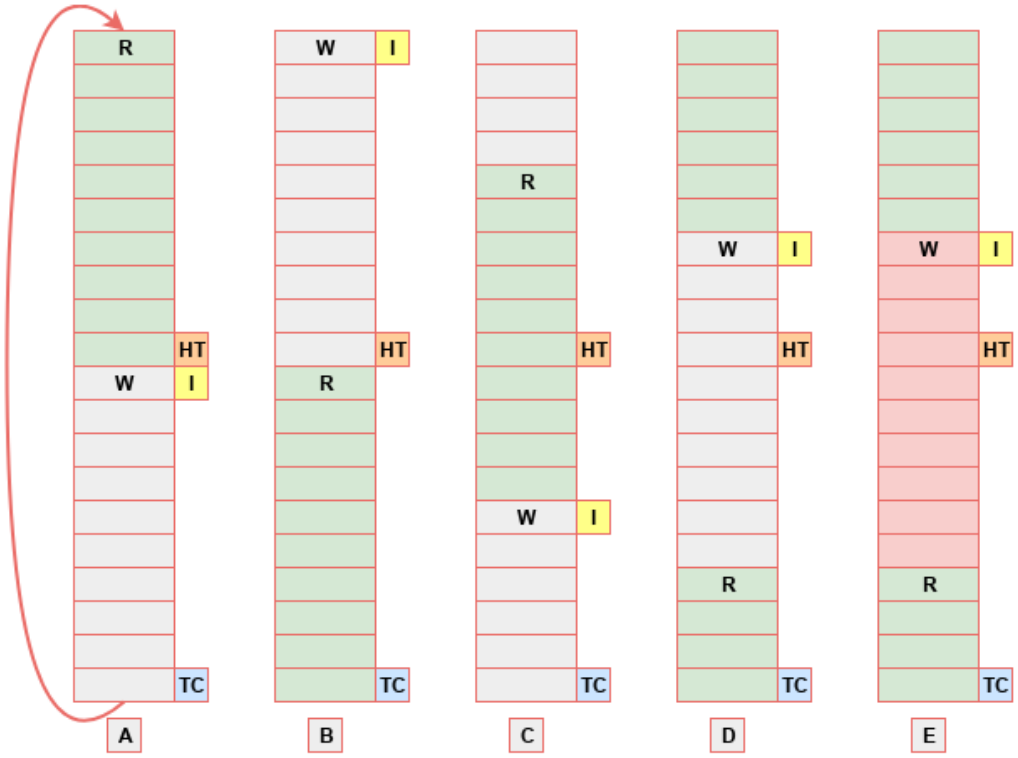


Рисунок 6.12 – Переривання DMA та UART при отриманні даних

Позначення на рисунку:

HT – переривання “половина буфера отримана”;

I – переривання “лінія UART неактивна”;

TC – переривання “буфер заповнений”

R – покажчик читання, що використовується програмою;

W – покажчик запису, що використовується DMA.

На рисунку 6.12 (E) показано приклад отримання 30 байтів даних без використання переривань HT та TC, використовуючи лише переривання I. В цьому випадку 10 байтів перезаписано і втрачено. Тому необхідно обробляти дані в процесі отримання, а не лише після завершення передавання, використовуючи переривання DMA (HT та TC).

Драйвер UART використовує запис в кільцевий буфер та чергу FreeRTOS для передавання оновлень позиції покажчика запису, які виникають при перериванні. Також відбувається повідомлення завдання (в даному проєкті Wi-Fi) для його розблокування у випадку, коли воно знаходиться в режимі очікування.

7.1.3 Структура завдань операційної системи

Діаграма класів завдань наведена в конструкторській документації на кресленику Д16. Діаграма класів із залежностями завдань наведена на кресленику Д17 (взаємні залежності між завданнями не показано). Діаграма послідовності роботи завдань наведена на кресленику Д18.

7.1.3.1 Драйвер WiFi

Зв’язок з Wi-Fi модулем ESP8266EX відбувається за допомогою інтерфейсу UART, драйвер якого описаний в розділі 66, з використанням протоколу AT-команд, або програмного набору Гаєса[79]. Повний командний набір описано в [81]. Враховуючи принцип асинхронності операції пристрою (тобто нова інформація або зміна стану з’єднання може відбутися в будь-який момент), необхідно виділити процес взаємодії та обміну даними з модулем ESP8266EX в окреме завдання операційної системи.

Блок-схема алгоритму роботи завдання наведена в конструкторській документації в кресленику Д7. Завдання знаходиться в стані очікування вхідної команди або повідомлення від модуля ESP8266EX. При отриманні даних від модуля вони розбиваються на окремі повідомлення та обробляються, змінюючи стан драйвера відповідно. Для обробки даних використано розроблену бібліотеку кільцевого буфера, що дозволяє працювати з даними в кільцевому буфері як з послідовним масивом. Алгоритм обробки даних, отриманих від модуля ESP8266EX наведено в кресленику Д8. Алгоритм обробки повідомлень наведено на кресленику Д9. У випадку отримання даних від ПК або зміни стану з'єднання завдання надсилає відповідний сигнал.

У випадку отримання команди від іншого завдання викликається процедура виконання відповідної команди. Алгоритм виконання команд наведений на кресленику Д10. Після обробки всіх команд надсилається відповідний сигнал, що дозволяє синхронізувати інші завдання. У випадку невдачі виконання завдання надсилається сигнал про помилку.

7.1.3.2 Завдання Log

В процесі роботи завдань виникає необхідність передати певні параметри роботи, телеметричні дані та інформацію про хід виконання програми або помилки на ПК. У зв'язку з ненадійністю мережі Wi-Fi може відбутися втрата зв'язку з мережею, втрата зв'язку TCP або втрата пакета. В такому випадку необхідно повторно передати втрачені дані. Для універсального забезпечення передавання даних від усіх завдань було розроблене завдання Log. Воно забезпечує з'єднання з ПК, повторне з'єднання у випадку втрати зв'язку, буферизацію даних та їхнє гарантоване доставлення. Алгоритм роботи завдання Log наведений в конструкторській документації на кресленику Д11.

7.1.3.3 Завдання Position

Завдання Position забезпечує позиціювання пристрою та імплементує алгоритм позиціювання, описаний в розділі 35. Блок-схема алгоритму завдання наведена в конструкторській документації на кресленику Д2. Для зв'язку з модулем давача положення використовується протокол I^2C та відповідний драйвер, описаний в розділі 63. На початку роботи завдання відбувається запис налаштувань в модуль LSM303C, після чого відбувається його калібрування, тобто визначення середнього статичного значення. Блок-схема алгоритму калібрування наведена на кресленику Д3. Після цього, при кожному перериванні про наявність даних, відбувається зчитування даних з внутрішнього буфера модуля (для зчитування використовується перший регістр даних, наступні регістри надсилаються модулем автоматично). Приклад обміну даними показано на рисунку 6.13[80], де:

SAD – адреса пристрою;

SUB – адреса внутрішнього регістра пристрою;

DATA – дані, що записуються або зчитуються;

ST – операція "Start";

SP – операція "Stop";

SR – операція “Repeated start”.

Table 12. Transfer when master is writing one byte to slave

Master	ST	SAD + W		SUB		DATA		SP
Slave			SAK		SAK		SAK	

Table 13. Transfer when master is writing multiple bytes to slave

Master	ST	SAD + W		SUB		DATA		DATA		SP
Slave			SAK		SAK		SAK		SAK	

Table 14. Transfer when master is receiving (reading) one byte of data from slave

Master	ST	SAD + W		SUB		SR	SAD + R			NMAK	SP
Slave			SAK		SAK			SAK	DATA		

Table 15. Transfer when master is receiving (reading) multiple bytes of data from slave

Master	ST	SAD+W		SUB		SR	SAD+R			MAK		MAK		NMAK	SP
Slave			SAK		SAK			SAK	DATA		DAT A		DATA		

Рисунок 6.13 – Обмін даними між давачем LSM303C та мікроконтролером

Аналогічно відбувається зчитування даних давача температури з модуля магнітометра, яке після зчитування записується в драйвер ультразвукового давача для розрахунку швидкости звуку.

Після отримання даних акселерометра відбувається їхнє усереднення та обробка відповідно до алгоритму. Блок-схема алгоритму обробки даних акселерометра наведена на кресленику Д4. Після обробки даних надсилається сигнал про наявність нових даних позиції.

7.1.3.4 Завдання DataProcessor

Завдання DataProcessor створено для обробки даних давача відстані та побудови карти перешкод, або растрової мапи імовірностей, за допомогою об’єднання даних з давача положення та давача відстані за математичною моделлю давача відстані та алгоритмом, описаним в розділі 48. Блок-схема алгоритму завдання наведена в конструкторській документації на кресленику Д5. Завдання очікує вимір ультразвукового давача, після чого розраховує відстань до перешкоди, отримує

положення та викликає алгоритм оновлення карти перешкод. Блок-схема алгоритму оновлення карти перешкод показана на кресленику Д6.

Завдання не надсилає готову карту перешкод на ПК, натомість надсилає дані положення та відстані, з яких розраховується відповідна карта. Це дозволяє зменшити кількість даних, що передаються, разом із тим повністю відтворивши карту перешкод, наявну на пристрої. Приклади побудованих карт перешкод наведені на рисунках 2.4 та 5.18.

7.1.3.5 Завдання Driver

Завдання Driver створене для незалежного керування двигунами, руху за поставленим маршрутом, зупинки в разі раптового виявлення перешкоди, дотримання заданої швидкости завдяки ПІ-регулятору та руху вздовж межі перешкоди у випадку потрапляння в локальний максимум, відповідно до алгоритму, описаного в розділі 55. Блок-схема алгоритму завдання наведена в конструкторській документації на кресленику Д12.

У випадку отримання команди руху вперед алгоритм послідовно отримує дані положення та аналізує напрямок руху автомобіля та напрямок руху до найближчої цілі згідно з маршрутом. У випадку відхилення від курсу повертаються передні колеса автомобіля для відповідного коригування напрямку. Алгоритм також враховує відстань до перешкоди та зменшує швидкість або зупиняється у випадку, коли відстань може призвести до зіткнення. При досягненні точок на маршруті руху передається відповідний сигнал до інших задач.

У випадку отримання команди слідування вздовж меж перешкоди алгоритм виконує послідовний рух “назад-вперед” для виїзду поза межі U-подібної перешкоди.

7.1.3.6 Завдання PathPlanner

Завдання PathPlanner керує рухом пристрою відповідно до заданої цілі, яка передається з ПК, відповідно до алгоритму, описаного в розділі 52. Завдання передбачає створення маршруту, його передавання завданню Driver, а також перемикання завдання Driver між режимами слідування по маршруту та вздовж межі перешкоди. Блок-схема алгоритму завдання наведена в конструкторській документації на кресленику Д13.

В процесі прямування до цілі завдання очікує на дані перешкод, після чого будує шлях за допомогою рекурсивної функції, що отримує точку початку та точку призначення і будує шлях між ними, розбиваючи його на два у випадку наявності перешкоди між точками. Блок-схема алгоритму побудови маршруту наведена на кресленику Д14.

7.2 Програма керування для ПК

Для візуалізації телеметричних даних, карти перешкод, а також керуванням руху автомобіля була розроблена система керування рухом автомобіля для використання на ПК. Код програми написаний мовою C++ з використанням графічної бібліотеки Qt. Код доступний в репозиторії[81]. Діаграма класів наведена в конструкторській документації на кресленику Д18.

7.2.1 Структура програми

Програма складається з головного вікна — об’єкта класу MainWindow, в якому розташовані елементи контролю та область візуалізації даних. Віджет SurroundArea використовується для візуалізації даних та отримання команд щодо руху автомобіля. Він використовує клас DataProcessor, аналогічний до використовуваного на пристрої, що гарантує побудову однакової карти перешкод.

7.2.2 Вигляд програми

Головне вікно програми зображене на рисунку 6.14. Область візуалізації даних (віджет SurroundArea) позначена (1). На ньому зображається візуалізація побудованої карти перешкод, де темніший колір означає більшу ймовірність, та історія положень автомобіля. При наведенні на комірку карти перешкод з'являється інформація про розраховану ймовірність перешкоди (рис. 6.15). При натисканні на область візуалізації координати натисненої точки передаються на пристрій як цільова точка.

Область статусу позначена (2), вона може набувати значення "Disconnected", "Waiting" та "Connected". Кнопка Start вмикає TCP-сервер для очікування підключення (рис. 6.16), після чого змінюється на кнопку Stop. В області повідомлень (4) показуються повідомлення від системи, які не є даними телеметрії.

Будь-які дані, які не починаються із заголовка одного з наборів телеметричних даних, описаних в розділі 76, інтерпретуються як користувацькі повідомлення та додаються в область повідомлень.

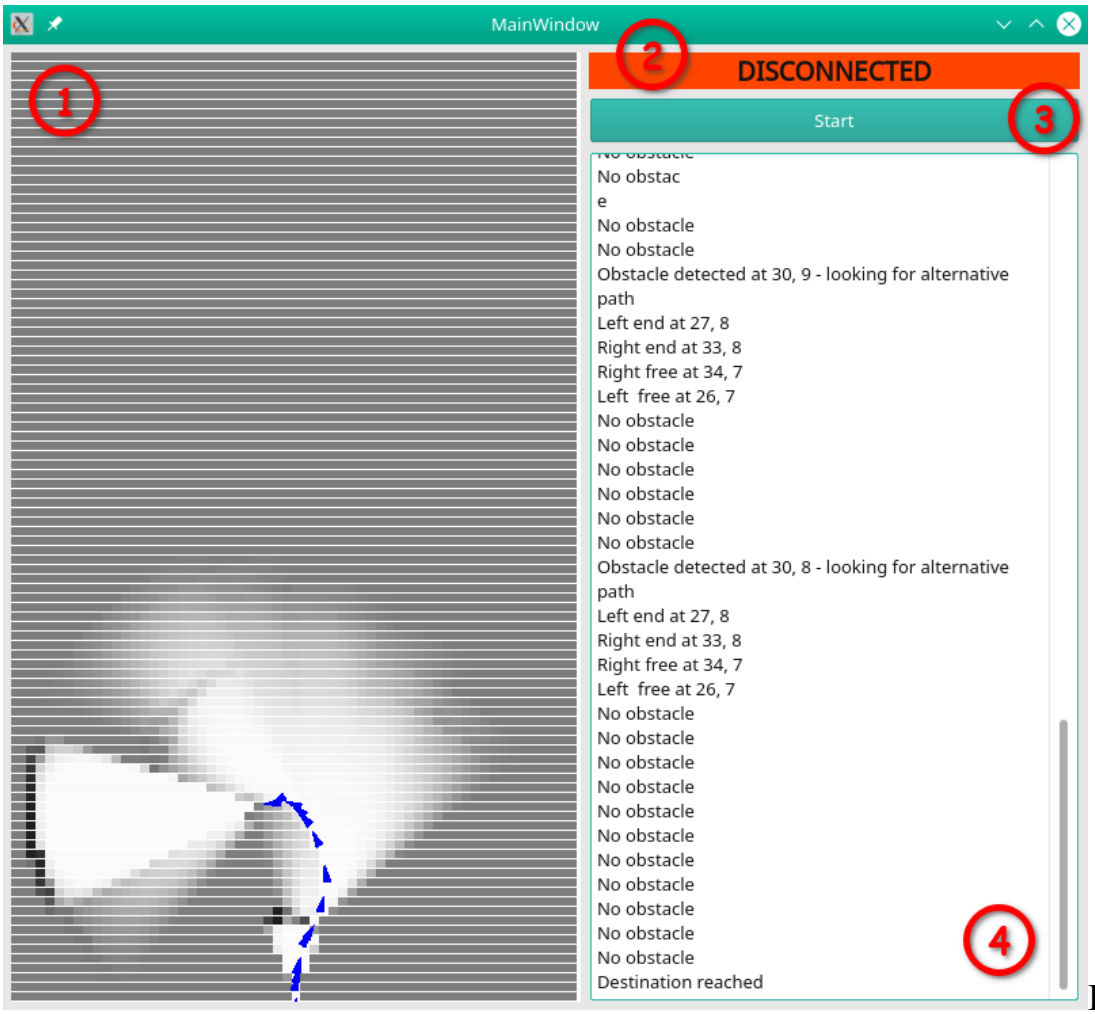


Рисунок 6.14 – *В*ікно програми керування для ПК

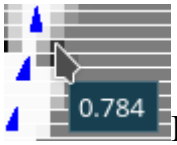


Рисунок 6.15

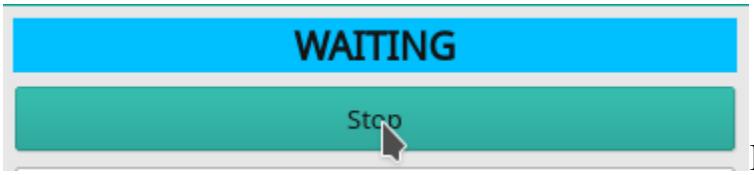


Рисунок 6.16 – Очікування на з’єднання

7.3 Протокол обміну даними

Для передачі даних телеметрії використовується наступний протокол передавання даних. Дані передаються в текстовому форматі, кожне повідомлення закінчується символом нового рядка (‘\n’). Телеметричні дані з пристрою на ПК мають бути відформатовані відповідно до таблиці 6.1.

Таблиця 6.1 – Формат даних телеметрії

Заголовок	**Роздільник**	**Дані**	**Роздільник**	**…**			**Кінець**
POS	:	0.0	,	0.0	,	0.0	\n
DST	:	0.0					\n

Заголовок визначає тип даних, що передаються. Дані розділяються комою та передаються у форматі дробових чисел з крапкою. POS – дані позиції, вони мають три аргументи — зміщення вперед, тангенційне зміщення та кут повороту пристрою. Всі відстані передаються в міліметрах.

Користувацькі дані, які не є даними телеметрії і надсилаються для інформування користувача, мають бути обов’язково розділені символом нового рядка.

Дані з ПК на пристрій передаються у вигляді пар координат (x,y), де числа передаються в цілому форматі в міліметрах та розділяються комою. Повідомлення закінчується символом нового рядка.

8 ВИСНОВКИ

В результаті виконання дипломного проєктування було розроблено модель автоматичної системи керування автомобілем. Обрано компоненти для її збірки та алгоритми роботи, побудовано архітектуру та розроблено програмне забезпечення системи, проведено прикладні випробування обраних алгоритмів.

Під час виконання завдання було проведено ґрунтовний аналіз наявних алгоритмів виявлення та ідентифікації перешкод, планування маршруту та руху з уникненням перешкод. Розроблено конструкцію пристрою, електричну схему та плату, під'єднано периферійні модулі давачів. До початкової схеми було додано модуль Wi-Fi, що дозволяє передавати телеметричні дані та отримувати команди з ПК.

Експериментально досліджено алгоритми позиціювання за допомогою акселерометра та магнітометра. Зроблено висновок про неможливість застосування магнітометра для визначення кута повороту в автомобілях з електродвигунами, оскільки електромагнітне поле, створене двигуном, набагато потужніше земного магнітного поля. Разом із тим, використання акселерометра для визначення лінійної швидкости та положення дає гарні результати. Використання акселерометра для вимірювання кута повороту можливе і дає прийнятні результати, проте для підвищення точности та уникнення дрейфу кута повороту при тривалому русі доцільніше застосовувати високоточні гіроскопи.

Було проаналізовано та експериментально досліджено алгоритми визначення перешкод та побудови ймовірнісної карти перешкод навколишнього середовища. На основі наявних алгоритмів було розроблено власний алгоритм побудови маршруту та руху по маршруту, що враховує кінематику пристрою та особливості використаних давачів. Використано пропорційно-інтегральний регулятор швидкости. Проаналізувавши динаміку руху автомобіля, зроблено висновок, що точність та якість роботи системи перш за все залежить від точности даних позиціювання.

Розроблені алгоритми та отримані результати досліджень можуть бути використані при проєктуванні різних типів самохідних роботів та інших рухомих пристроїв, наприклад домашніх роботів, роботів на підприємствах, безпілотних автомобілів та роботів для дослідження інших планет.

Розроблена архітектура та побудована програмна система керування, що складається з двох частин — системи керування автомобілем та програми для візуалізації телеметричних даних (положення та перешкод) в реальному часі та визначення цілі, що виконується на ПК. Розроблено протокол передавання даних між системою керування та програмою на ПК.

Отримані результати повністю відповідають завданню на дипломне проєктування.

9 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Obstacle Detection Techniques in Outdoor Environment: Process, Study and Analysis / Y. Singh, L. Kaur. // International Journal of Image, Graphics and Signal Processing. – 2017. – №5. – С. 35–53.

2 Generic Obstacle Detection Method for Collision Avoidance / N. Ben Romdhane, M. Hammami, H. Ben-Abdallah. // IEEE Intelligent Vehicles Symposium. – 2011.

3 Wikipedia [Електронний ресурс] – Режим доступу до ресурсу: https://www.wikipedia.org/.

4 From Google to Tesla, it's a War of LiDAR or RADAR [Електронний ресурс] // UnitedLex. – 2017. – Режим доступу до ресурсу: https://www.unitedlex.com/news/from-google-to-tesla-its-a-war-of-lidar-or-radar.

5 Performance comparison of Infrared and Ultrasonic sensors for obstacles of different materials in vehicle/ robot navigation applications / S. Adarsh, S. M. Kaleemuddin, D. Bose, K. I. Ramachandran. // IOP Conference Series: Materials Science and Engineering. – 2016. – №149.

6 Menezes P. Low Cost Sensor Based Obstacle Detection and Description / P. Menezes, J. Dias, H. Araújo, A. de Almeida. – Coimbra: Instituto de Sistemas e Robótica.

7 Elfes A. Using occupancy grids for mobile robot perception and navigation / Alberto Elfes. // Computer. – 1989. – №22. – С. 46 – 57.

8 Ribeiro M. I. Obstacle Avoidance / Maria Isabel Ribeiro. – Lisboa: Instituto Superio Técnico, 2005. – 14 с.

9 McGuire K. A Comparative Study of Bug Algorithms for Robot Navigation / K. N. McGuire, G. C. H. E. de Croon, K. Tuyls, 2018. – 22 с.

10 Bug Algorithms and Path Planning / Steven Roderick et al. – University of Maryland.

11 Kamon I. TangentBug: A Range-Sensor-Based Navigation Algorithm / I. Kamon, E. Rimont, E. Rivlin. // The International Journal of Robotics Research. – 1998. – Т. 17, № 9. – С. 934–953.

12 An Autonomous Sensor-Based Path-Planner for Planetary Microrovers : Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C) / S. L. Laubach, J. W. Burdick. – 1999. – С. 347–354.

13 Xu Q. Vectorization path planning for autonomous mobile agent in unknown environment / Q. Xu, G. Tang. // Neural Comput & Applic. – 2013. – №23. – С. 2129–2135.

14 Rajamani R. Vehicle Dynamics and Control / Rajesh Rajamani — Springer US, 2006. – С. 20–27.

15 STM32F401xB STM32F401xC Datasheet – ST Microelectronics, 2019. – 139 с.

16 Product User's Manual – HC-SR04 Ultrasonic ranging sensor.

17 I^2C-bus specification and user manual – NXP Semiconductors, 2014. – Rev. 6 – 64 с.

18 LSM303C Ultra-compact high-performance eCompass module: 3D accelerometer and 3D magnetometer Datasheet – ST Microelectronics, 2014. – Rev. 2. – 53 с.

19 ESP8266EX Datasheet – Espressif Systems, 2020. – Rev 6.4. – 30 с.

20 Seifert K. Implementing Positioning Algorithms Using Accelerometers / Kurt Seifert, Oscar Camacho — Freescale Semiconductor, 2007. – 13 с.

21 Larnder C. I. A purely geometrical method of determining the location of a smartphone accelerometer / Christopher Isaac Larnder. – 10 с.

22 Elfes A. Sonar-Based Real-World Mapping and Navigation / Alberto Elfes. // IEEE JOURNAL OF ROBOTICS AND AUTOMATION. – 1987. – Т. RA-3, №3. – С. 249–265.

23 Feedback Control of Computing Systems / J. L. Hellerstein, Y. Diao, S. Parekh, D. M. Tilbury. – Wiley-IEEE Press, 2004. – 456 с.

24 FreeRTOS documentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.freertos.org/.

25 Description of STM32F4 HAL and LL drivers / ST Microelectronics, 2017. – 1838 с.

26 Репозиторій коду проєкту [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/ukrkyi/CarRider.

27 General-purpose timer cookbook for STM32 microcontrollers / ST Microelectronics, 2019. – 72 с.

28 STM32F401xB/C and STM32F401xD/E advanced Arm®-based 32-bit MCUs Reference manual / ST Microelectronics, 2018. – 847 с.

29 Majerle T. STM32 tutorial: Efficiently receive UART data using DMA [Електронний ресурс] / Tilen Majerle. – 2017. – Режим доступу до ресурсу: https://stm32f4-discovery.net/2017/07/stm32-tutorial-efficiently-receive-uart-data-using-dma/.

30 Majerle T. STM32 UART DMA RX/TX [Електронний ресурс] / Tilen Majerle – Режим доступу до ресурсу: https://github.com/MaJerle/stm32-usart-uart-dma-rx-tx.

31 ESP8266 AT Instruction Set / Espressif Systems, 2019. – Версія 3.0.2. — 68 с.

32 Репозиторій коду програми для ПК [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/ukrkyi/CarControl.

33 Оформлення текстових документів у навчальному процесі. Стандарт організації (кафедри) СОУ АУТС 01-16. Для студентів кафедри автоматики та управління в технічних системах [Текст] / Уклад.: Я.Ю. Дорогий, Н.Б. Репнікова, О.І. Ролік, Л.Ю. Юрчук – К.: НТУУ «КПІ», 2016. – 27 с.